CONTRIBUTIONS TO NEUROMORPHIC AND RECONFIGURABLE CIRCUITS AND SYSTEMS

A Thesis
Presented to
The Academic Faculty

by

Stephen H Nease

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
August 2011

CONTRIBUTIONS TO NEUROMORPHIC AND RECONFIGURABLE CIRCUITS AND SYSTEMS

Approved by:

Professor Paul Hasler, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor David Anderson
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Maysam Ghovanloo
School of Electrical and Computer Engineering
Georgia Institute of Technology

Date Approved: June 29, 2011

ACKNOWLEDGEMENTS

First, I would like to thank my advisor, Dr. Paul Hasler. I appreciate his patience with me as I stumble towards becoming a researcher. I'm always amazed by his ability to talk with students in detail for hours about a plethora of subjects relating to the lab's efforts. Probably most important is his confidence in the work of his students. He firmly believes in our capabilities and states so frequently. Having that support from an advisor is invaluable.

I also sincerely thank all of my labmates for so kindly helping me with this thesis. They've taught me enormous amounts about the finer points of all sorts of topics I had no idea even existed two years ago. While their technical help has been useful, they have taught me so much more than nuts and bolts. They have showed me by example how to perform good research in a deliberate, consistent manner while always considering the fundamental principles of the problem at hand.

Finally, I would like to thank my family. They have contributed more than anyone else to my advancement towards this degree. Ever since I first expressed interest in this field, they have been one hundred percent behind me. When I doubt my abilities, they remind me of past successes and give me motivation to push through difficult times. Their confidence in me is sometimes the only thing that keeps me going. This document would not exist without their support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

This work presents three projects which I undertook in the field of reconfigurable and neruomorphic circuits and systems. In a way, they chart the progress of my knowledge about the subjects over the past two years.

The first chapter introduces the concept of a Field Programmable Analog Array (FPAA), in much the same way as it was presented when I first took ECE 6435 with Dr. Hasler. I describe the primary components of one particular FPAA, the RASP 2.8a. Building upon that foundation, the next chapter presents an application of this knowledge to circuits. I implemented and characterized a neuromorphic circuit which behaves like a dendrite.

After understanding the RASP 2.8a and implementing a circuit on it, the next step in my education was to actually make part of an FPAA. The third chapter describes this process. I designed, laid out, and tested part of an FPAA whose job was to perform current-mode computation.

The final chapter in my education was to learn about a complex neuromorphic system. I helped in the testing of a new system, a portion of which I report in the fourth chapter. I created software which removes offsets in part of the system.

CHAPTER I

INTRODUCTION

1.1 Description of this work

Over the past few years, the field of reconfigurable analog signal processing has yielded many interesting opportunities for research. Analog has enjoyed a renaissance recently: as the sizes of embedded systems shrink, their power requirements become more stringent. Digital systems are typically more power-hungry than analog, so many of their functions are being replaced by analog frontends. The problem with creating analog frontends is twofold. First, they are expensive to design, both in time and money. Second, they suffer from transistor mismatch, which is an especially difficult problem when transistors are operated in their subthreshold regime, where their currents are exponentially dependent on gate voltage.

A solution to both of these problems is the Field Programmable Analog Array (FPAA). The FPAA is an analog CMOS chip that contains many general-purpose analog elements which can arbitrarily be connected together using a matrix of floating-gate pMOS switches. This concept has a digital equivalent in Field-Programmable Gate Arrays (FPGAs). The reconfigurable nature of the FPAA means that a single chip can be used for hundreds of different applications, so there is no need to completely redesign the chip from the ground up every time a new system is needed. Since the FPAA uses floating-gate transistors, mismatch among analog elements can be eliminated by programming these transistors to a desired state.

Additionally, the reconfigurable analog mindset can be applied to a burgeoning field known as neuromorphic engineering. The basic goal of the field is to create artificial systems in CMOS which behave similarly to real neural systems in biology.

Since the brain is the most power-efficient processor known, the hope is that emulating biology will create extremely low-power and robust computational systems. It is well-known that neurons are complex dynamical systems, and analog CMOS is the perfect substrate for creating such dynamics. As it turns out, many of the physical principles behind the operation of neurons and silicon systems are very similar.

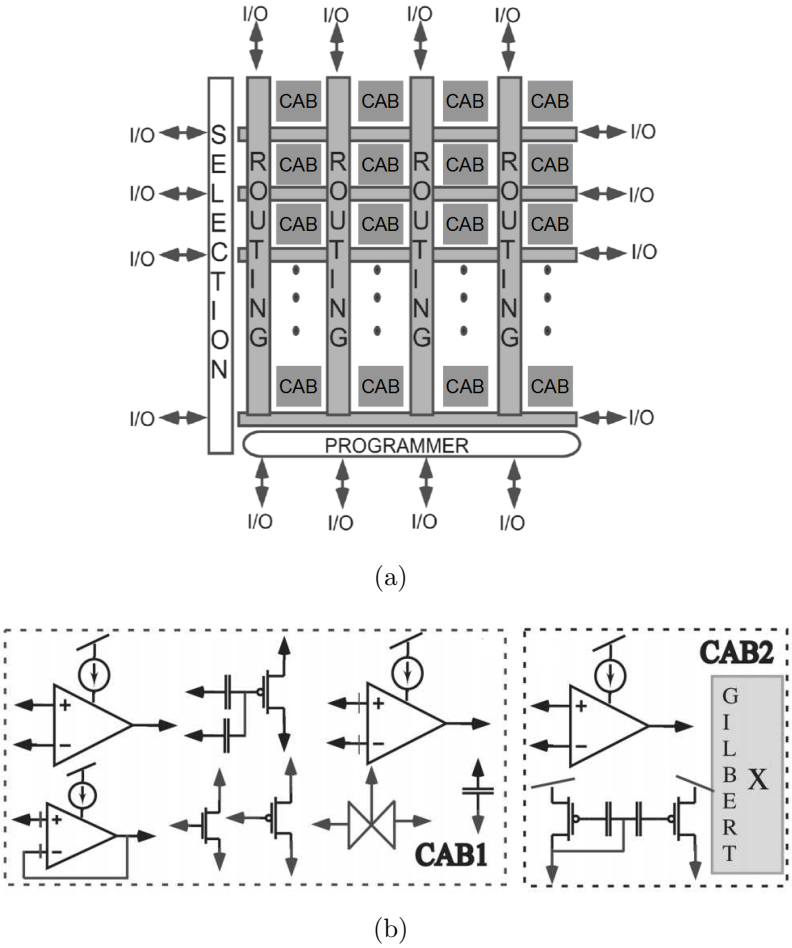
This thesis describes work conducted in a few different areas of reconfigurable analog processing and neuromorphic engineering. The first project could be considered a primary example of the usefulness of the FPAA. A completely generic RASP 2.8a FPAA was used to implement a neuromorphic dendrite circuit. The benefits and drawbacks to using a generic FPAA are discussed, as well as the similarity of the results to biology. The second project involved the design, layout, and testing of a subsystem for a completely new FPAA, the RASP 2.9v. The final project involved using floating-gate programming to remove offsets in a neuromorphic FPAA, the RASP Neuron 1D. These three projects have provided an invaluable introduction to the field of reconfigurable analog signal processing, and this base will be built upon to create even more complex and useful systems in the PhD dissertation.

1.2 Introduction to the FPAA

All of the data presented in the first chapter comes from a reconfigurable hardware platform that can be used to develop neuromorphic models. The FPAA is a mixed-signal CMOS chip which allows analog components to be connected together in an arbitrary fashion, allowing for rapid testing and measurement of many different circuit designs. The specific chip used for that chapter is the RASP 2.8a [1].

The FPAA is organized into three functional blocks, which are shown connected together in Fig. 1(a). The first is the Computational Analog Block (CAB), which is a collection of analog circuits which act as computational elements. These elements include nFETs, pFETs, Operational Transconductance Amplifiers, capacitors, Gilbert

multipliers, and others. A schematic representation of the two available CABs in the RASP 2.8a is shown in Fig. 1(b). These CAB components can be connected together to form more complicated subcircuits, which can be further interconnected to create an analog computational system.



(a)

(b)

Figure 1: (a) Block diagram of the RASP 2.8a. (b) Schematic of available CAB components. Both images modified from [1].

The interconnection of CAB components is accomplished with the FPAA's second functional block, the switch matrix. This is a collection of switches which connects together rows and columns of routing lines. The routing lines connect I/O lines and CABs. A schematic of the switch matrix interconnection scheme on the RASP 2.8a is shown in Fig. 2(a). The elements of the switch matrix are floating-gate pFET (FG-pFET) switches. A floating-gate pFET has essentially the same structure as a

traditional pFET, except that its gate has no DC path to ground. Voltage is applied to the gate through a capacitive divider. The lack of a DC path to ground means that once charge is stored on the gate, it will remain there without the need for a directly-applied potential. We are able to place charge on the gate and remove charge from it using the quantum mechanical processes of Fowler-Nordheim tunneling and hot electron injection. Since we are able to place an arbitrary amount of charge at the gate, the pFET devices can pass both high and low voltages, similar to a transmission gate. The switch resistance of a FG-pFET is shown in Fig. 2(b). Its nominal value of 10 kΩ results in negligibly small voltage drops when subthreshold currents are used for the circuits.

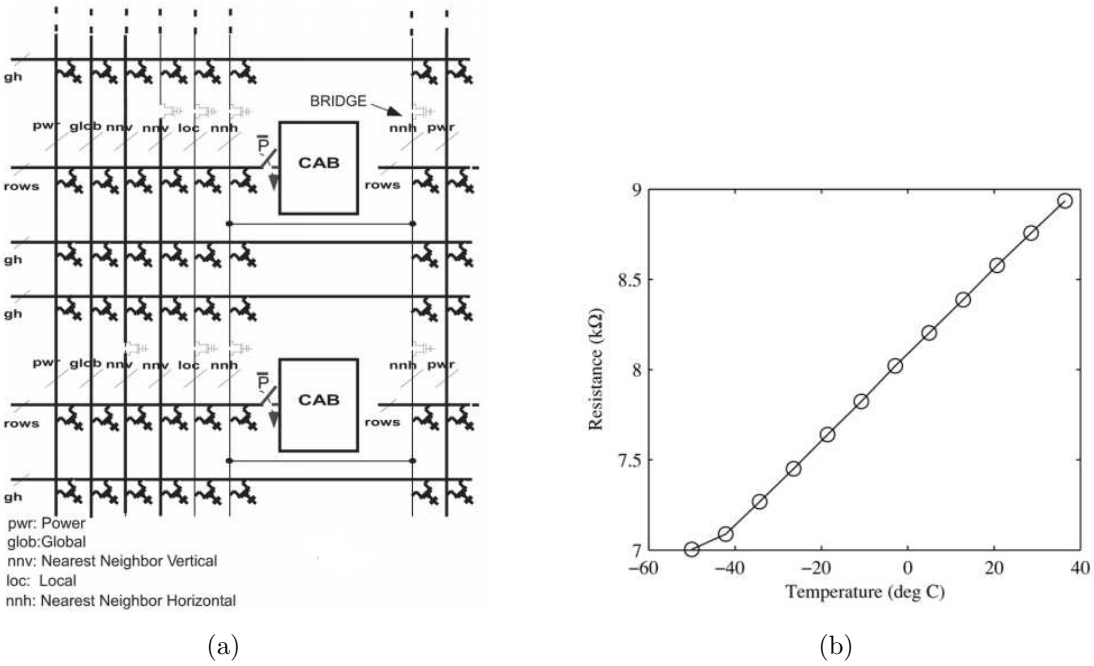


Figure 2: (a) Switch matrix connections on the RASP 2.8a. (b) Typical switch resistance. Both images modified from [1].

The third functional block is the programmer, which selects a floating-gate device in the switch matrix and controls the processes of tunneling and injection to add or remove charge from the floating gate. A basic schematic of the programmer is shown in Fig. 3. The first stage of programming is the "erase" stage, where the floating-gate

voltage is reset to a high value using forward and reverse Fowler-Nordheim tunneling. This involves putting a high electric field across the oxide in the device to allow electrons to travel across the oxide barrier, which changes the charge on the floating node [14].

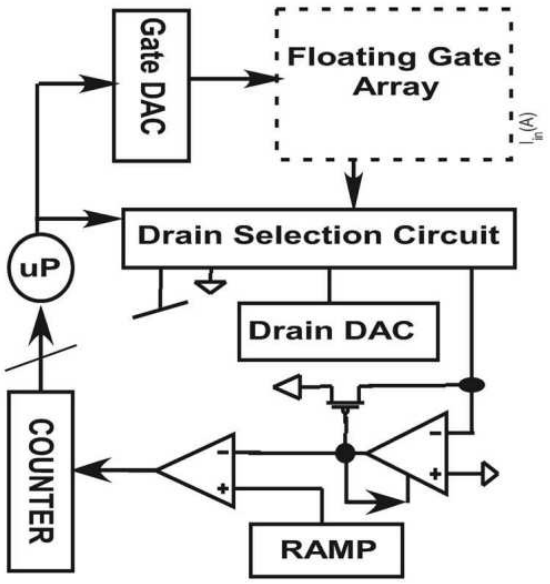


Figure 3: Programmer for the RASP 2.8a. Image from [1].

The next phase is the program/measure cycle. A feedback loop is formed around the floating-gate, and the programmer alternates between reducing the floating-gate voltage and measuring the result. Hot-electron injection reduces the floating-gate voltage by running a high current through the device, allowing electrons to overcome the potential barrier and inject into the floating-node. This adds negative charge to the node and hence lowers its floating-gate voltage. Injection is alternated with measurement of the floating-gate voltage. During the measurement phase, the FG-pFET's drain is connected to a current measurement circuit. The current through the FG-pFET for a uniform set of terminal conditions will change based on the floating node charge, so measuring current is equivalent to measuring floating-gate voltage. In Fig. 3, the current measurement circuit is a logarithmic TIA ramp ADC, but we also use a MOSFET diode for I-to-V conversion. The value of the current is sent

to MATLAB through a microcontroller. Based on the measured current, injection is performed repeatedly and at different rates to get the floating-gate voltage to its target.

The programming infrastruture allows each device to be turned completely on, completely off, or operated somewhere in-between. This flexibility means that switch elements can be used for computation as well as routing, a benefit seen in other efficient routing applications [24], [13]. One example of a useful computational element created from floating-gates is a constant current source.

Now that we have introduced the basic concepts of a reprogrammable hardware platform, the following three chapters will discuss the three projects undertaken in this thesis. The final chapter will conclude with a summary of the contributions made and commentary on the future direction of research.

CHAPTER II

MODELING AND IMPLEMENTATION OF VOLTAGE-MODE CMOS DENDRITES ON A RECONFIGURABLE ANALOG PLATFORM

2.1 The Neuromorphic Engineer's Thesis: Silicon Emulates Biology

Neuromorphic engineering has garnered ever-increasing interest ever since Carver Mead's early explorations of the field [16]. Neuromorphic engineers claim that transistors can be used to emulate biological processes. Silicon devices and biological structures operate based on similar physical principles, so it is possible to make circuits which share many of the computational properties of neurobiological systems. There are two consequences of this statement: neuromorphic circuits can be used to natively simulate biological systems, and they can also be used to perform bio-inspired computation. This paper explores how neuromorphic technology can be applied towards emulation of dendritic behavior.

Computational neuroscientists use mathematical models implemented on digital computers to simulate biological processes. While these are powerful tools, their effectiveness is decreased as simulation sizes grow. However, neuromorphic engineering promises a different paradigm: simulation through the physics common to silicon technology and biological systems. This allows for real-time emulation of dense biological systems, rather than a slower and less efficient numerical simulation.

While faster simulations of biology is one benefit of neuromorphic devices, they can also perform useful computation. One important structure in biology is the dendrite, a highly-branched conductive medium which connects the neuron's synapses to its

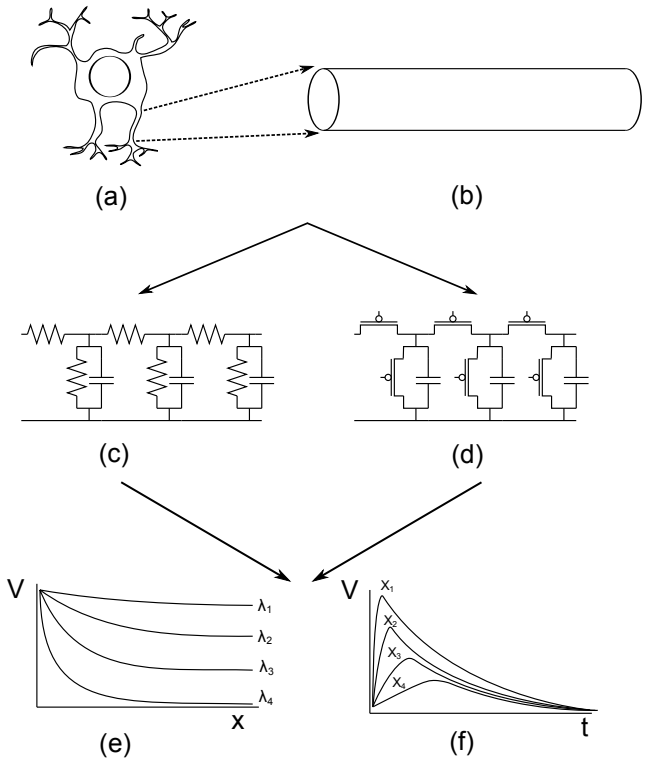


Figure 4: Illustration of the connections between biological dendrites and CMOS structures. Both biology and these CMOS circuits demonstrate the steady-state and dynamic properties of linear cables.

soma, as shown in Fig. 4. When operated in the correct regime, a VLSI dendrite model produces the behavior predicted by canonical linear models. Dendrites are the structures which connect synapses to the cell body. They perform linear (and sometimes nonlinear) summations of input currents (4a). Neuroscientists typically model these structures as passive linear cables (4b). The classical model for this linear cable is an equivalent RC delay line. The major predictions of linear cable theory are based on this model (4c). An alternative model for the linear cable is a network of aVLSI elements, primarily MOSFETs and capacitors, where input currents are translated into small voltage signals which swing around a DC operating point ((4d)). If (c) and (d) are equivalent, they should behave similarly. The steady-state behavior of both models is expected to be an exponential decay in voltage, where the amount of decay depends on physical parameters (4e). The dynamic behavior of

both models is expected to be exponential decay in space and a delay in time (4f).

For many years, neuroscientists assumed that dendrites did not add much significant computational value to networks in the brain. They were modeled simply as wires by researchers in artificial neural networks. Recently however, a number of neuroscience reviews have posited that dendrites and single neurons have more computational power than previously believed [11], [15]. In order to begin to take advantage of this computation, we have verified that some of the most basic properties of dendrites can be observed using analog CMOS circuit models.

There is a long history of dendritic emulation in the neuromorphic community. One of the first projects was undertaken by Elias [5], who created compartmental models consisting of resistors of various layouts, capacitors, and synapses implemented by MOSFETs. He demonstrated spatial weighting of inputs, sublinear summation of nearby synapses, and tonic summation of inputs. He implemented simple computational systems such as a signal symmetry detector and a direction selectivity system.

Another classic dendritic implementation was completed by Rasche and Douglas [19]. They chose to implement the axial conductances with switched capacitors and the leakage conductances with OTAs. They performed extensive tests on how cable properties change as a function of the conductances, boundary conditions, and compartment lengths. They also observed how action potentials propagated down the passive cable and demonstrated that the cable could act as a directionally sensitive system.

A more recent review was completed by Wang and Liu [25]. Their system consisted of computational subunits which included nonlinear synapses, a spiking circuit, and a cable connecting the compartments together. They investigated how the response of the dendrite changes based on the spatiotemporal pattern of the inputs to the system. They emulated N-methyl-D-aspartic acid (NMDA) channels, which are

ligand-gated channels for the neurotransmitter glutamate [10]. They showed how activating NMDA channels leads to superlinear responses in the system and that these nonlinearities allow the dendrite to discriminate between input patterns with different spatial extents.

This chapter's primary focus is to discuss the benefits and drawbacks of a simple dendrite model implemented in a reconfigurable environment. Our model consists of P-channel MOSFETs operating in their linear regime. We choose this topology because it lends itself well to scaling in a reconfigurable environment. We run simple experiments that demonstrate the model's equivalence to a classical linear cable, and then we discuss nonidealities caused by the reconfigurable environment.

This chapter is organized as follows. We begin by discussing the fundamental unit of computation in neuromorphic systems – the silicon channel – and state that it can be used to model biological channels. Next, we discuss how we have used this platform to bias silicon channels in a way that simulates the voltage-mode behavior of dendrites, as described by Rall's linear cable model. We then discuss tools our lab has developed to aid in the design of dendritic circuits. Finally, we provide a brief overview of extensions of the basic cable model, such as nonlinear behavior of the circuits and implications of placing them in large-scale analog systems.

2.2 The Silicon Channel

Neuromorphic engineering begins with the principle that the transistor acts as a biological analog. Carver Mead recognized that this is true because both silicon and biological channels behave according to the same natural principle. The channel of a transistor operated in its subthreshold regime is governed by the diffusion equation, as are many biological processes [16].

The physical structure of a MOSFET consists of polysilicon, silicon dioxide, and doped n-type silicon. The channel of a transistor is a region of silicon that separates

the drain from the source (see Fig. 5a). This area forms an energy barrier to charge carriers at the source and at the drain. The number of charge carriers at the source or drain end of the channel is determined by the size of this barrier, which is modulated by the difference between the gate voltage and the source or drain voltage. Since the source is operated at a higher potential than the drain in the P-channel device, the barrier at the source end of the channel is lower, so there are more charge carriers at the source end of the channel than at the drain end. Therefore we have a gradient of charge carriers from the source end of the channel to the drain end. This is illustrated in Fig. 5c. This means that carriers must diffuse from the source to the drain according to the diffusion equation from [16]:

$$v_{diffusion} = -D\frac{1}{N}\frac{dN}{dh} \tag{1}$$

where $v_{diffusion}$ is the velocity of carriers, D is the diffusion constant, N is the number of charge carriers per unit volume, and h is distance. When the diffusion equation is applied in the case of a gradient of charge carriers from the source to the drain of a pFET channel, the current is given in [14] as:

$$\begin{aligned} I &= I_0 e^{\kappa(V_{dd}-V_g)/U_T}\left(e^{-(V_{dd}-V_s)/U_T} - e^{-(V_{dd}-V_d/U_T)}\right) \\ &= I_0' e^{-\kappa V_g/U_T}\left(e^{V_s/U_T} - e^{V_d/U_T}\right) \end{aligned} \tag{2}$$

V_{dd} is the well potential of the pFET, V_g is the gate voltage, V_s is the source voltage, and V_d is the drain voltage, all referenced to ground. I_0 is a collection of physical constants which is intuitively the saturation current when $V_g = V_s = V_{dd}$. κ is a measure of how well the gate voltage modulates the potential at the channel's surface. U_T is the thermal voltage (typically around 26 mV at room temperature). To simplify the nomenclature, we can reference the terminal voltages to V_{dd}, in which case $I_0' = I_0$. To reference everything to ground, we let $I_0' = I_0 e^{\kappa V_{dd}/U_T} e^{-V_{dd}/U_T}$.

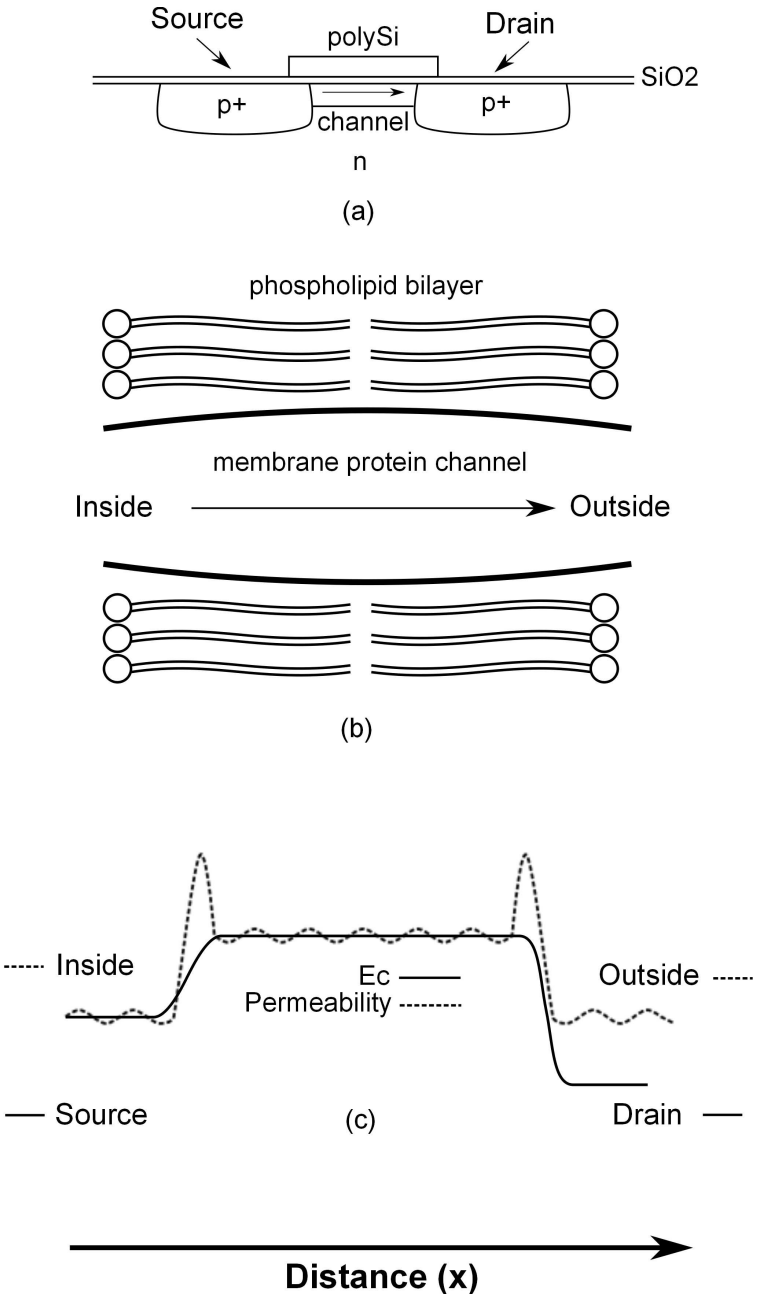


Figure 5: Illustration that biological and silicon channels share similarities.

The idea of overcoming energy barriers to produce current is also seen in biological channels. In Fig. 5b, we show the structure of a channel embedded in a membrane. The physical structure of a biological channel consists of an insulating phospholipid bilayer and a protein which stretches across the barrier. The protein is the channel in this case. Fig. 5c shows how both biological and silicon channels generate barriers to current, where the barrier is potential energy in the case of silicon and permeability in the case of biology. The band diagram of silicon (solid line) has a similar shape to the classical model of membrane permeability proposed by Danielli [9] (dashed line).

2.3 Implementing the Linear Cable Model with Analog CMOS Circuits

Historically, dendrites have been modeled as linear cables. Their structure consists of a conductive solution that allows current to flow from the synapse to the cell body; a phospholipid bilayer which separates the membrane potential from the external potential; and ion channels which allow small amounts of current to leak across the membrane. Wilfrid Rall adapted the mathematics originally developed to model core conductor cables and applied it to dendrites [22]. We wish to demonstrate that the behavior of a CMOS dendrite with pFET channels reduces to Rall's mathematical model when operated with small-signal inputs.

Our basic thesis is shown in Fig. 6. A biological dendrite is modeled as a conductive cylinder surrounded by an insulating layer. A cross section of this model is shown in Fig. 6a, where I_{ax} represents the current flowing along the axial direction of the dendrite, I_{Lk} represents current from the dendrite to extracellular fluid through a leak channel, and the internal and external potentials are V_{mem} and E_k, respectively. When we translate channels into transistors, we get the model shown in Fig. 6b, where both the axial and leakage current flow through transistors. The external voltage is set by a voltage source E_k, and V_{mem} is set by the bias structure. When we linearize the transistor model, the result is shown in Fig. 6c and Fig. 6d. Current

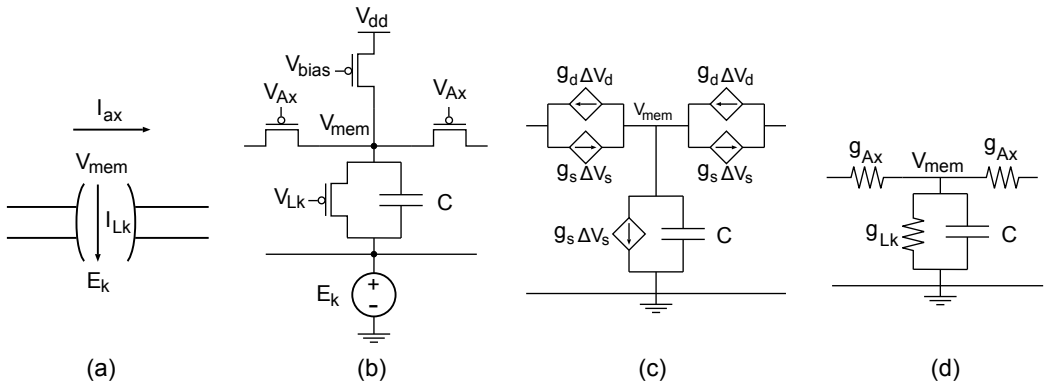


Figure 6: Demonstrating the biological (a), CMOS (b), and small-signal (c and d) models of a dendrite.

sources can be reduced simply to small-signal conductances.

2.3.1 Introduction to Linear Cable Theory

The simplest model neuroscientists use to describe the function of dendrites is known as the Linear Cable Model. The dendrite is treated as a conductive core surrounded by an insulating layer. The core is modeled as a long piece of resistive material, which can be discretized into many incremental resistances R_{Ax}. The insulating layer is a phospholipid bilayer, and it is modeled as a capacitance C because it separates the internal membrane potential from the extracellular potential. However, there is leakage current from the intracellular solution to the outside of the cell, so a leakage resistance R_{Lk} is also included in the model.

Koch gives a simple derivation of the mathematical cable model for this circuit in [10]. If one writes down Kirchhoff's Current Law (KCL) at the nodes V_{mem} and uses Ohm's Law $V = IR$ and the capacitor equation $I = C\frac{dV}{dt}$, then the following differential equation describes the system:

$$\lambda^2 \frac{\partial^2 V_{mem}}{\partial x^2} = \tau \frac{\partial V_{mem}}{\partial t} + V_{mem} - R_m I_{inj} \tag{3}$$

where I_{inj} is current injected into the dendrite, $\tau = R_{Lk}C$ and $\lambda = \sqrt{\frac{R_{Lk}}{R_{Ax}}}$. τ and λ are called the time constant and the space constant. Intuitively, τ determines how

voltages along the dendrite change with time, and λ determines how voltages change with distance down the dendrite. If we only care about the steady-state solution, we can set the differential with respect to time equal to zero. This results in a solution for the steady-state behavior given in Eq. 4.

$$V(x) = V_0 e^{-|x|/\lambda} \tag{4}$$

2.3.2 Using Silicon Channels to Implement the Linear Cable Model

Our goal is to replace the resistances in the linear cable model with silicon channels. The most intuitive way to do this is to simply replace each resistance with a single pFET. The axial resistances are replaced with a pFET whose gate is set at a fixed potential, V_{Ax}. Similarly, the membrane resistances are replaced with pFETs whose gates are set at a fixed potential V_{Lk}. On an intuitive level, the conductance of the pFETs is set by their gate voltage. We will need to bias the dendrite at a fixed membrane potential, so a transistor which provides a DC bias current is inserted into each node of the dendrite. It has a gate voltage V_{bias}, and it sets the DC point V_{mem}. The final piece of the dendrite to consider is the capacitance. It is a fact of analog circuits that every node has some capacitance associated with it. So we do not have to place an explicit capacitance at each node to simulate a dendrite. If we so desire, the FPAA has the ability to compile 500 fF capacitances into the nodes. The final circuit is as shown in Fig. 6b.

In order to model an equivalence to the linear cable model, we can simplify the full circuit into a linear one. Each transistor is replaced with a small-signal, linearized model. To do this, we take partial derivatives of the current equation for a pFET as formulated in Eq. 2.

2.3.2.1 Linear Model of Axial FET

In the operation of the circuit, we will leave the gate fixed at a DC bias, so we can simplify Eq. 2 by incorporating the gate voltage term into $I_{bias} = I_0' e^{-\kappa V_g / U_T}$. Therefore, the current through the axial and leakage pFETs can be expressed as follows:

$$I = I_{bias} \left(e^{V_s/U_T} - e^{V_d/U_T} \right)$$

Traditionally, we form a linear model for this device by taking the partial derivative of the current with respect to a changing terminal voltage. Since a signal is traveling in the axial direction of our dendrite, both the source and the drain of the axial FET are changing. We model this with two current sources in parallel pointing in opposite directions, with the values $g_s \Delta V_s$ and $g_d \Delta V_d$. Ignoring channel length modulation, the values for g_s and g_d are given in [14] as:

$$g_s = \frac{\partial I_{Ax}}{\partial V_s} = \frac{I_{bias}}{U_T} e^{V_s/U_T}$$

$$g_d = -\frac{\partial I_{Ax}}{\partial V_d} = \frac{I_{bias}}{U_T} e^{V_d/U_T}$$

Note that, at rest, the dendrite will be biased such that all source and drain nodes of the axial pFETs will be at the same rest potential, V_{rest}. This means that $g_s = g_d$. We can combine the two current sources into one source with the value

$$\begin{aligned} I &= g_{Ax} \Delta V_s - g_{Ax} \Delta V_d \\ &= g_{Ax} \left(\Delta V_s - \Delta V_d \right) \\ &= g_{Ax} \Delta V_{sd} \end{aligned}$$

So this is simply a small-signal conductance,

$$g_{Ax} = \frac{I_{bias_{Ax}}}{U_T} e^{V_{mem}/U_T} \tag{5}$$

2.3.2.2 Linear Model of Leakage FET

Modeling the leakage transistor is much easier. Both the gate and the drain are fixed to DC voltages. So any change in voltage across the device is completely due to a change in the source. Therefore, the small-signal conductance of the leakage FET is just the source conductance, as given above:

$$g_{Lk} = \frac{I_{bias_{Lk}}}{U_T} e^{V_{mem}/U_T} \tag{6}$$

2.3.2.3 Deriving the Space and Time Constants

The space constant is the parameter λ in the linear cable equation which describes how voltage in the dendrite decays with position along the dendrite. It is related to the ratio of the axial and leakage conductances. Now that we have linearized our model, we can define a space constant λ by taking the ratio of our conductances:

$$\begin{aligned} \lambda &\equiv \left(\frac{R_{Lk}}{R_{Ax}}\right)^{1/2} = \left(\frac{g_{Ax}}{g_{Lk}}\right)^{1/2} = \left(\frac{I_{bias_{Ax}}}{I_{biasLk}}\right)^{1/2} \\ &= e^{\frac{\kappa(V_{Lk}-V_{Ax})}{2U_T}} \end{aligned} \tag{7}$$

Figure 7 verifies this expression experimentally using the FPAA. We measured how the conductance of a pFET changes as a function of its DC gate potential. We took a CAB pFET and measured a reference source conductance by fixing the DC potential at all of its terminals (V_{s_0}, V_{g_0}, and V_{g_0}), and measuring the DC current. We then swept its source voltage through a very small range (V_{sweep}) and measured the change in current. The reference conductance was the slope of change in current with respect to change in source voltage. We performed this same experiment for

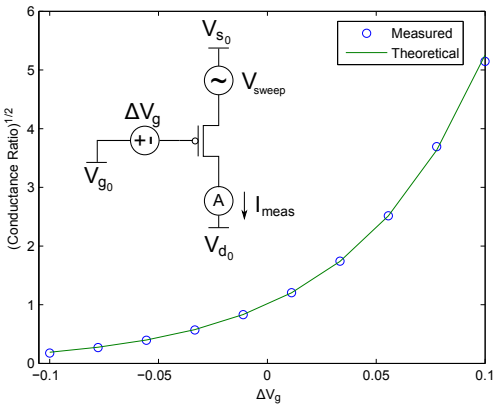


Figure 7: Experimental demonstration that the ratio of source conductances is a function of the difference between gate voltages.

Table 1: Sources of error on the RASP 2.8a

	μ	σ
κ	0.8393	0.0021
I_0	4.5740 fA	0.77549 fA

ten different values of the gate voltage (V_{g_0} - ΔV_g). We then plotted the square root of the ratio of source conductances as a function of the gate voltage. We used the difference in gate voltages to create a theoretical value of the conductance ratio from Eq. 7, and the two match very closely.

The time constant τ describes how voltages decay with time. It is defined as the product of the leakage resistance and the capacitance, or

$$\tau = \frac{C}{g_{Lk}} = \frac{CU_T}{I_{bias_{Lk}}} e^{-V_{mem}/U_T} \tag{8}$$

2.3.2.4 Sources of Error

The above expressions hinge on perfect matching among all pFET devices. This unfortunately is rarely achieved. We measured the values of κ and I_0 for a sample of 15 pFET CABs in the FPAA and measured the statistical variation for these two parameters. This information is shown in Table 1:

The above analysis assumes the system is processing "small" signals. We can no longer assume that the linear models behave if they are perturbed far from the DC bias. We limited inputs to the system such that the source nodes of the vertical pFETs never changed by more than 25 mV.

2.4 Demonstrating Equivalence to the Linear Cable Model

We now wish to demonstrate that our voltage-mode circuit retains many of the behaviors of a passive dendrite. We set up our cable using the system shown in Fig. 8(a). Each block representing a stage consists of one bias, axial, and leakage transistor as shown in Fig. 6b. At the output of each stage, two amplifiers relay the signal to a mux. The first is an open-loop floating-gate OTA which is used to measure step responses at each stage of the dendrite. The second amplifier is a buffer-connected OTA which is used to accurately read DC voltages for steady-state experiments.

2.4.1 Steady-State Experiments

The first test to perform is a steady-state analysis. In our experiment, we compiled a 10-stage dendrite onto the FPAA. We set $E_k = 1V$ and biased the membrane voltage to around 20 mV above E_k. Due to mismatch among the bias transistors and leak transistors, not all membrane voltages were exactly the same, and they could vary by as much as tens of mV. We attempted to compensate for some of the mismatch by an iterative process of measuring and changing the bias voltages on the gates of the I_{bias} transistors, but this did not remove all of the mismatch.

For five different values of λ, a small DC current was injected into the first node. We measured $\Delta V_i = V_{mem_i} - V_{rest_i}$ for every node in the dendrite. Then ΔV_i was normalized. The dots are experimental measurements, and the lines represent how the voltages should decay if λ matches the theoretical value perfectly. The theoretical values essentially predict the "slope" of the logarithmic response, and not the actual DC offset. This is why the normalized predictions are accurate for low values of

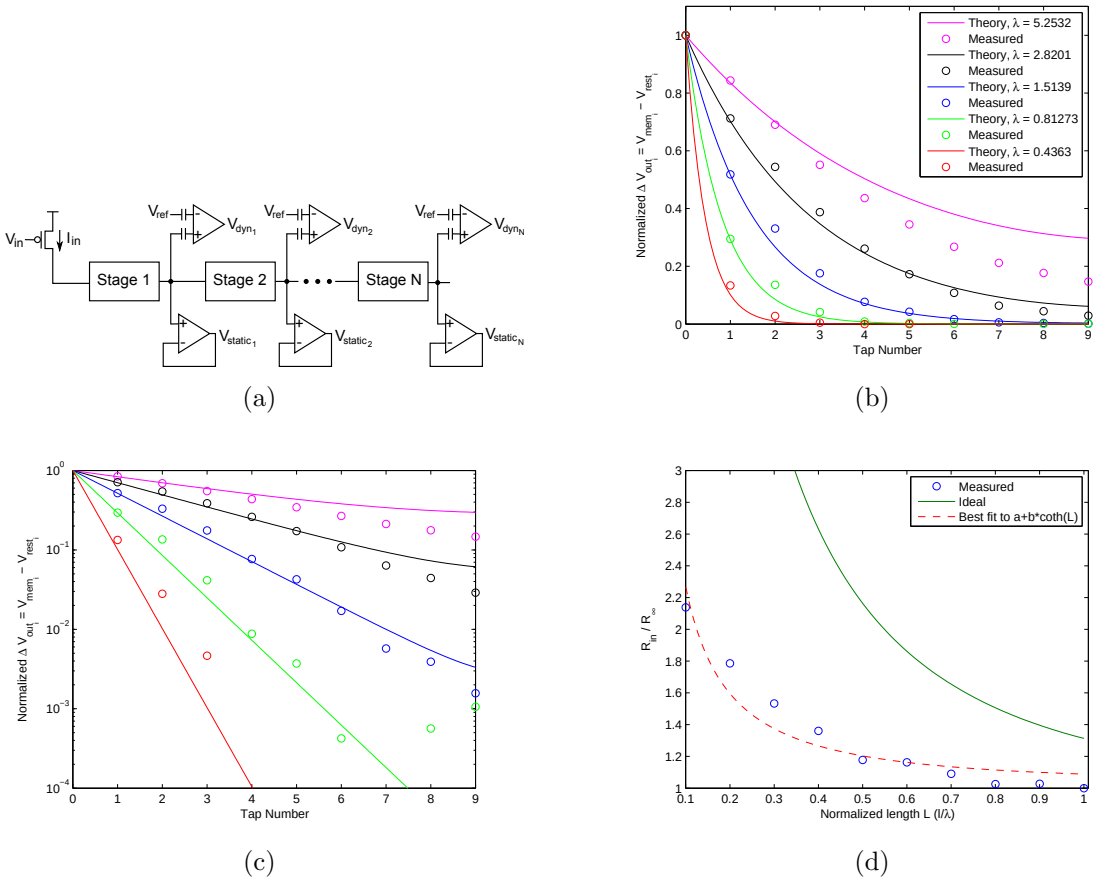


Figure 8: Setup and results for steady-state dendrite experiments.

stage number and seem to deviate as stage number increases. We're seeing error in the slope but not DC offset. The linear plot gives an intuitive physical feel for how the dendrite behaves, while the logarithmic plot demonstrates how these are approximately exponential responses. The log plot also shows how error in slope accumulates. Please note that any changes which were negative (all of which were small) are not shown on the log plot.

Since this is a dendrite of finite length, the steady-state solution takes on a slightly different form than that given earlier. From [10], the solution is

$$V(X) = V_0 \frac{\cosh(L - X)}{\cosh(L)} \tag{9}$$

where X = x/λ and L = l/λ. For this experiment, we defined the steady-state voltage

of a particular node as the difference between its measured rest voltage and its voltage after applying an input. The results for this dendrite are given in Fig. 8 b,c.

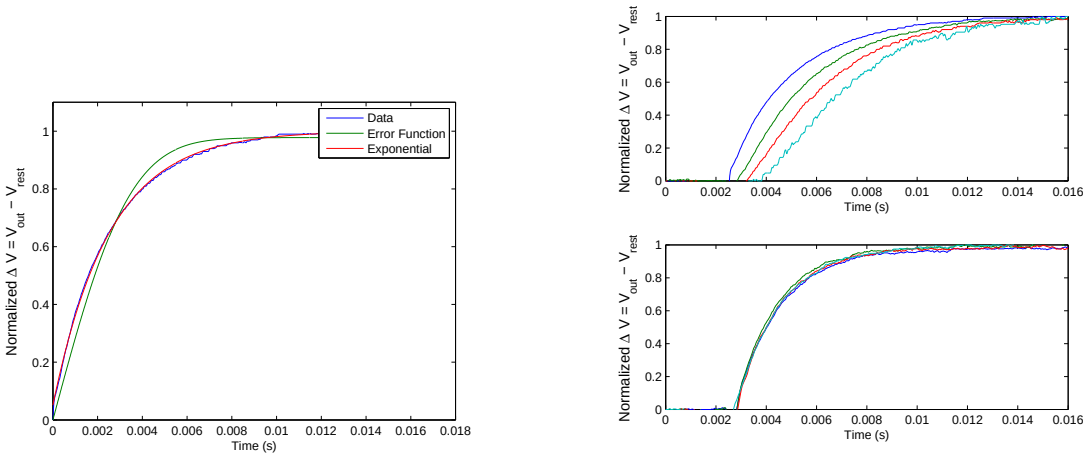
The input resistance of a semi-infinite, sealed-end cable is also well-known. Its expression is given in [10] as

$$R_{in} = R_{\infty} \coth(L) \tag{10}$$

As L increases, R_{in} approaches R_{∞}.

To test whether our dendrite follows this model, we applied a step input current of I_0 to our dendrite and varied the value of λ. A fixed input current was injected into Node 1 of the diffusor, and the membrane voltage at that node was measured before and after injection. We then calculated the difference between these two ($V_{delta} = V_{mem} - V_{rest}$). This was done for many different dendrite lengths. To calculate R_{in}/R_{∞}, we divided all values of V_{delta} by the value for L = 1. Since the injected current was the same for all tests, the ratio of resistances is therefore the ratio of the voltage responses. Our results are shown in Fig. 8(d). The response did not follow the quantitatively predicted curve, but it does demonstrate qualitative behavior similar to what we expect, as shown by the dashed curve, which is a curve fit to a + b*coth(L)

Our theoretical results do not perfectly match the data, and there are a few possible reasons for this. Probably the largest contributor to the problem is biasing the dendrite correctly. For the experiments in Fig. 8 b,c, the resting membrane potentials were as much as 30 mV away from each other. The ratio of small-signal conductances is $e^{(\Delta V/U_T)}$, so this means that the ratio of two ideally matched conductances could be as high as 3.32. It should also be noted that κ changes with the source voltage, so a 30 mV mismatch in source voltage could also affect κ.



(a) (b)

Figure 9: Dynamic experimental results. Fig. 10 shows parasitic transients not visible in this figure.

2.4.2 Dynamic Experiments

Cable theory provides us with a prediction for what the shape of the step response should look like at the site of current injection. The form is given in [10] as

$$V_{Step}(0,T) = \frac{I_0 R_\infty}{2} erf(\sqrt{T}) \tag{11}$$

We have plotted a representative step response for x=0 along with a best-fit line to this theoretical function in Fig. 9a. The step response was obtained by setting the value of V_{ref} on the first node's floating-gate OTA such that V_{dyn_1} was midrail. Then the input current was pulsed, and the waveform was captured. We experimentally determined how much to pulse V_{in} by alternatively pulsing it, measuring how much the first node's voltage changed, and adjusting the gate until the first node's voltage changed by less than U_T, or 25 mV. We chose this value since the FETs would leave saturation if the source voltage changed by much more. We normalized the result by subtracting the DC offset and dividing by the maximum value reached. Linear cable theory predicts that the error function will be a closer fit than the exponential, but the data for our system mirrors an exponential response much more closely. It is

possible that our step size was greater than needed to keep all devices in their linear regimes.

Since the cable model is basically an RC network, we expect to see delay down the line. The propagation velocity of a step input down the line is given in [10] as

$$v = 2\frac{\lambda}{\tau} \tag{12}$$

This means that we can increase the delay down the line (decrease the velocity of propagation) by decreasing λ or increasing τ. In our experiment, we changed λ and looked at how the velocity of propagation was affected. The results are shown in Fig. 9b. For a small value of λ, the velocity of propagation is small, so one can see delays of the response as they travel down the dendrite. For higher values of λ, the velocity of propagation is very fast, so very little delay can be seen. Note that Fig. 10 shows parasitic transients not visible in this figure.

In both the steady-state and dynamic experiments, we have seen a trend in our results. Namely, they agree with cable theory qualitatively but do not match it precisely, quantitatively. We do not expect these nonidealities to affect usability of the dendrites greatly. This is because we believe the computation in dendrites is not governed by precise tuning of every parameter. Neural computation is inherently different from the von-Neumann architectures in which precision is key. They exhibit high levels of stochastic behavior, redundancy, and recurrent connections. Rather, for us it was more more important to see that the basic dendritic properties can be varied over a wide range, allowing gross tuning of parameters.

2.4.3 Effects of a Reconfigurable Testbed

A reality of working in a reconfigurable environment is that parasitics can cause nonidealities to crop up when experiments are run. Fig. 10 demonstrates this. Two parasitic effects seen at once for one particular step response. Recall that the input

to the system is a pFET whose gate is being pulsed. When the gate of the pFET is pulsed down, some of that voltage change is coupled into the input node of the dendrite, and therefore initially the voltage at the membrane decreases. This change can be seen propagating along the system. For this step response, we also see a spike upwards. This is likely due to capacitive coupling into the instrumentation amplifier (a floating-gate input OTA), because this change is not seen propagating down the dendrite in other plots.

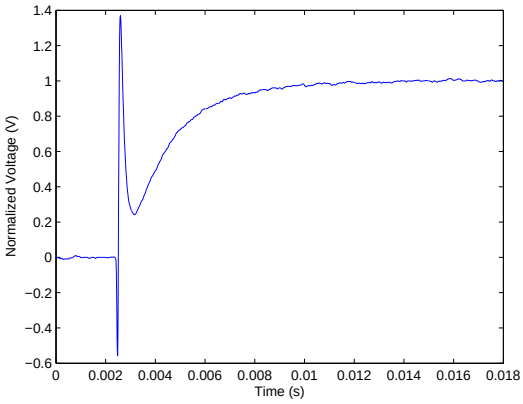


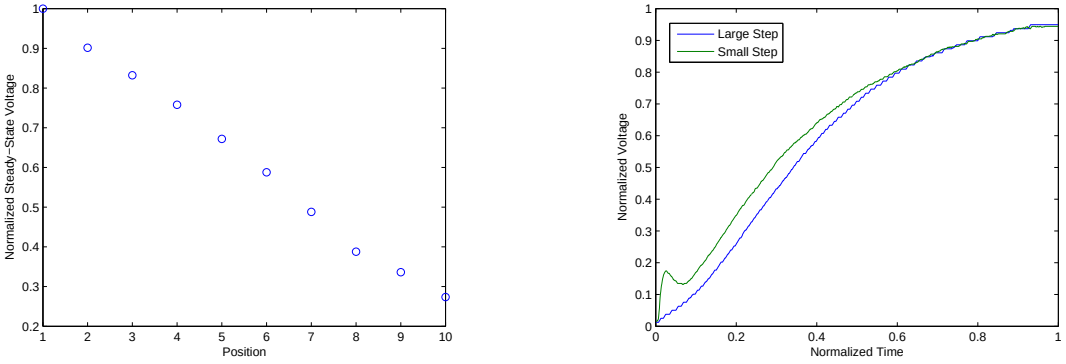
Figure 10: Nonidealities in the results of dendritic experiments due to FPAA parasitics.

The amount of coupling depends on how the system is routed, so certain care should be made to ensure that system components are routed to minimize such effects. For instance, the routing lines for the voltage measurement circuitry should not be physically close to the digital pulse on the gate of the input current source. Additionally, a cascode should be used on the input current source.

2.5 Nonlinear Behavior of Dendrites

Most of this paper has concerned the behavior of the dendritic circuit operated in its linear regime. When the input current becomes large, however, the qualitative behavior of the circuit changes, and nonlinear effects begin to take hold. Typically, a difference between drain and source of about $4U_T$, or 100 mV is typically considered

the nonlinear regime of the dendrite. In order to get a qualitative understanding of the nonlinear effects, we will analyze one "section" of dendrite, shown in Fig. 12.



(a) (b)

Figure 11: Nonlinear responses in the dendritic circuits.

When the steady-state response of a 10-stage dendrite is measured with a large input current (causing a change of about 200 mV at the first node), the response is a linear degradation in voltage (Fig. 11(a)). Fig. 11(b) compares the shapes of small step and large step response. The step response was normalized in voltage by dividing by the steady-state value, and time was normalized by finding the point at which the voltage rises to 95% of its steady-state value. The initial response of the small step is more of an RC response, while the large step shows a sigmoidal behavior. See Fig. 10 for a discussion of the transient at the beginning of the small step.

2.5.1 Math Modeling

Applying KCL and the current equations for a capacitor and a saturated transistor,

$$\frac{dV_s}{dt} = \frac{I_{in}}{C} - \frac{I_{bias}}{C} e^{V_s/U_T} \tag{13}$$

We can use Eq. 13 to plot a phase portrait. The basic shape is a negative exponential with a vertical offset, shown in Fig. 13.

This portrait gives us quantitative and qualitative information about our circuit's

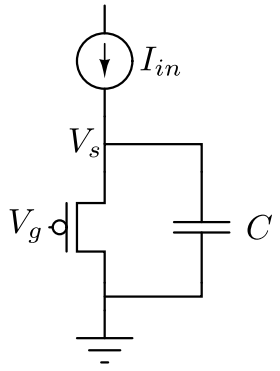


Figure 12: Illustration of nonlinear dynamics in dendrite circuit. A large-signal input current is sent into a node which sees a transistor and capacitor in parallel.

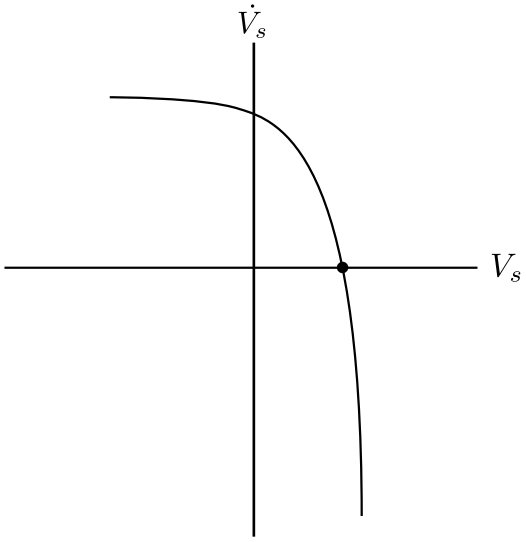


Figure 13: Illustration of the phase portrait resulting from the circuit in Fig. 12. The input current moves the line vertically, which changes the qualitative behavior of the system.

voltage response to an input current. First, it gives us the voltage where we expect V_s to settle:

$$V_s = U_T \ln \frac{I_{in}}{I_{bias}} \tag{14}$$

Second, the picture tells us that we will get small time constants for large values of I_{in}. Note from Eq. 13 that the vertical offset of this plot is determined by the value of I_{in}. As I_{in} increases, the plot is shifted up, and the rate at which V_s changes for a given value of V_s will be increased, thus decreasing the time constant. It is also important to point out that the slope of the actual phase portrait is much steeper than what we drew in Fig. 13. This means that a shift up in the plot won't affect the steady-state value of V_s as much as it will affect the time constant.

2.5.2 Demonstration of Impact on Dendrite Circuit Behavior

If we apply a large enough input current such that the membrane voltage changes by more than 100 mV, we can measure the effects of nonlinear input currents on the dendrite.

Our first experiment was to see how the steady-state voltage decays, as shown in 11a. The result is that the voltage decays linearly with space. This is a desirable effect, since it is essentially a compression operation. Recall that, for small inputs, the steady-state voltage decayed exponentially. If this trend were to continue for large inputs, the dynamic range of available voltages would be severely limited. However, for a large input, the FETs are no longer operating as resistors; they are in saturation, so we merely require linear changes in voltage to achieve exponential changes in current. Therefore the dendrite is using nonlinearity to increase its dynamic range.

Our second experiment is to see how the shape of the step response changes with an increase in input current. We can rewrite Eq. 15 in the current domain. Defining $I_1 = I_{bias} e^{V_s/U_T}$, we can differentiate with respect to time to get $\dot{I}_1 = I_1/U_T \dot{V}_s$.

Substituting into Eq. 13,

$$I_{in} = \frac{CU_T}{I_1}\frac{\partial I_1}{\partial t} + I_1$$
$$\frac{\partial I_1}{\partial t} = \frac{I_1}{CU_T}\left(I_{in} - I_1\right) \tag{15}$$

When Eq. 15 is solved, it behaves like a tanh function, so we expect the shape of our dendrite's step response to be sigmoidal for large current steps. Our results in Fig. 11 bear this out.

2.6 Implementing Dendrites in Large Reconfigurable Systems

Recall that the FPAA connects analog components together using a matrix of floating-gate pFET switches. These FG pFETs can be used as regular transistors, as well, so they can be connected to form floating-gate diffusors. Rather than explicitly apply a gate voltage to the horizontal and vertical transistors, they can be programmed with varying levels of charge, which effectively acts like an applied voltage. The switch matrix must by design be an extremely dense array of switches, so we can make very large dendrites as inputs into neurons.

2.6.1 Difficulties of Floating-Gate Diffusors

Modeling floating-gate denditic circuits is more complicated than with regular FETs. The reason for this is that the capacitive coupling from the source and drain to the floating-gate is more pronounced than with regular pFETs. In order to design a floating-gate dendrite, therefore, an extra step of characterizating these coupling ratios is necessary. If we desire more complicated behavior by programming different values of the floating-gate voltage for different sections of the dendrite (i.e. changing the dendrite's diameter), we will need to take these coupling ratios into account when determining to what voltage we want to program the floating gate. We need

to know coupling ratios because floating-gate transistors are programmed with their terminal voltages at one potential (in "program mode"), and after programming their terminal potentials undergo a change (in "run mode," when the circuit is operating). An example of a floating-gate diffusor not behaving as expected is shown in Fig. 14.

The simplest way to characterize the capacitive coupling is to perform sweeps of each terminal and extract an "effective κ" for that terminal. This measure tells how much a change in one terminal voltage will modify the floating-gate potential. Then if we have a desired membrane potential, we know how much all of the terminals will change in the transition from program mode to run mode, and we know how the floating-gate voltage will be affected. Once we know that floating-gate voltage, we can attempt to program the bias transistor to match the current it is drawing. More than likely, we will require an iterative process of programming the bias transistors, measuring the membrane voltage, and reprogramming to achieve the desired membrane potential.

Another important nonideality in floating-gate systems which requires characterization is the fact that the transistor which is programmed differs from the transistor which is actually placed in the circuit. This scheme is known as indirect programming, and any differences between the programmed and in-circuit transistor will affect the circuit's performance. Methods to characterize these effects are discussed in [23].

2.6.2 Benefits of Floating-Gate Diffusors

The most exciting aspect of dendritic circuits is that they can be made in an extremely compact manner. As we stated above, the switch matrix of the RASP 2.8a FPAA is completely made up of floating-gate switches. So there is potential to make huge arrays of dendrites using the switch matrix. Since the purpose of the array is to interconnect components, it makes sense that dendrites be used to send signals from one compiled structure to another. Fig. 15 is an example of how such a diffusor might

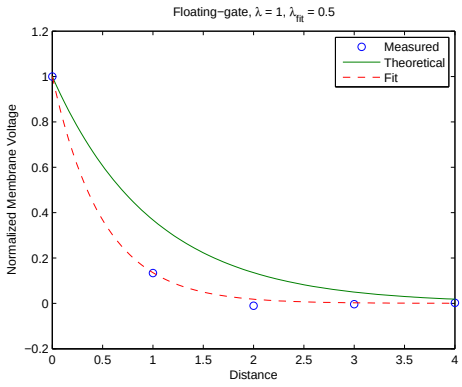


Figure 14: Illustration of offsets introduced by capacitive coupling from the drain of the diffusor.

be made. Partitioning of the switch matrix allows for a large number of dendrites to be created.

In a switch matrix, a floating-gate transistor exists at every intersection of two wires which can short a horizontal line and vertical line. In the representation of Fig. 15, an intersection with a black dot represents wires which have been shorted together with a floating-gate transistor. A picture of a transistor represents a floating-gate which is part of the diffusor structure and is programmed somewhere between open and closed circuit. No graphic at an intersection represents a floating-gate which has been programmed open-circuit. The leftmost column has been shorted to ground, and all the transistors connected to it are the vertical devices in the diffusor. The rightmost column has been shorted to V_{DD}, and all the transistors connected to it are the biasing devices. The pairs of two floating-gates in the middle are the horizontal transistors which connect the vertical legs together.

We can estimate how large these dendrites can be based on the FPAA routing structure. Each CAB has an associated floating-gate switch matrix. Some rows and columns are global, meaning they have connectivity among multiple CABs. We will only consider local rows and columns which do not connect beyond a CAB. In addition, the columns have semi-local connections to their nearest vertical and

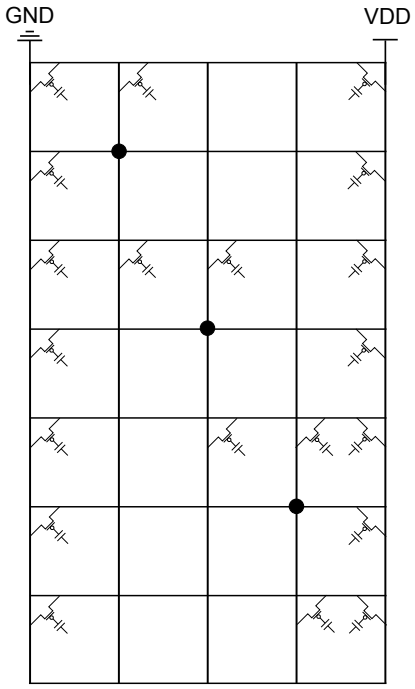


Figure 15: Possible method of placing dendrite in switch matrix.

horizontal neighbors, so we assume that half of those columns are available per CAB. The equivalent number of useful columns per CAB is 14. The rows are hard-wired to CAB elements, so the number of usable rows is reduced to ensure no CAB devices are turned on. For CAB types 1 and 2, the number of available rows is 24 and 34, respectively. If we make a dendrite as shown in Fig. 15, each row connects to one vertical transistor, and each column connects to two horizontal transistors. The size of dendrites in CAB type 1 is limited by its number of rows, while CAB type 2 is limited by columns. Therefore, we estimate that CAB types 1 and 2 can implement dendrites of approximately 24 and 28 stages, respectively. Based on the numbers of these CABs in the FPAA, we can theoretically make 28 dendrites of length 24 and 4 dendrites of length 28. We can then use the global routing to chain some of these together.

It is also important to point out that neural systems are inherently imprecise. Real synapses are very unreliable, and no two dendritic structures are the same. So the

disadvantages listed above are not necessarily detriments. Some amount of variability from dendrite-to-dendrite caused by floating-gate transistor mismatch could be seen as a good thing. In fact, the inability to precisely model the behavior could be an asset, for it requires designers to get an intuitive feel for what parameters work well for a given system.

2.7 Conclusion

We have demonstrated that it is possible to use an FPAA to create a voltage-mode CMOS dendrite which maintains certain properties of linear cables. We have seen qualitative behaviors that are similar in both the steady-state response and the dynamic response. With this as a fundamental building block in neuromorphic circuits, we are now free to explore more interesting topologies.

The next step in this line of research is to demonstrate computational primitives using these dendritic structures. Simple dendritic computations have been proposed for a long time, such as coincidence detection and simple boolean operations caused by local inhibition [11], [15]. These computations are often supported by active channels [25]. For example, a recent paper showed that both the passive properties of linear cables and the nonlinear effects of NMDA channels cause dendrites to respond more strongly to a centripetal sequence of inputs than a centrifugal sequence [2], [4]. These simple computations have the potential to form more powerful units. Specifically, we have proposed that they could be used to aid HMM-like classification [8], and we are working towards demonstrating this functionality experimentally. We hope to leverage these units for useful classification and discrimination systems.

CHAPTER III

DESIGN, LAYOUT, AND TESTING OF A COMPUTATIONAL ANALOG BLOCK FOR CURRENT-MODE DIGITAL-TO-ANALOG CONVERTERS

Proponents of analog signal processing believe that analog systems should act as the frontend for many applications since they offer low-power solutions to simple computations. However, some problems are better-suited for digital systems, such as control, memory, and high-precision computation. So digital systems are often needed to accompany these analog frontends.

An important topic in reconfigurable systems research is how to communicate information between analog and digital subsystems. The RASP 2.9v is a new FPAA with enhanced capabilities for interfacing with digital systems. This work involved the design, layout, and testing of a portion of the RASP 2.9v: a Computational Analog Block whose purpose is to act as a reconfigurable current-mode Digital to Analog Converter (a "DAC CAB"). In the following sections we will briefly introduce the major features of the 2.9v chip, describe the design considerations for the DAC CAB, and then show experimental results from a current-mode DAC compiled on this chip.

3.1 Introduction to the RASP 2.9v

An overview of the intended use of the RASP 2.9v is shown in Fig. 16. The CADSP lab has developed software which allows users to create Simulink blocks which represent components of the CAB. These blocks are translated into SPICE netlists by

a program called Sim2Spice, and these netlists are in turn translated into lists of floating-gate targets by a program known as GRASPER [12].

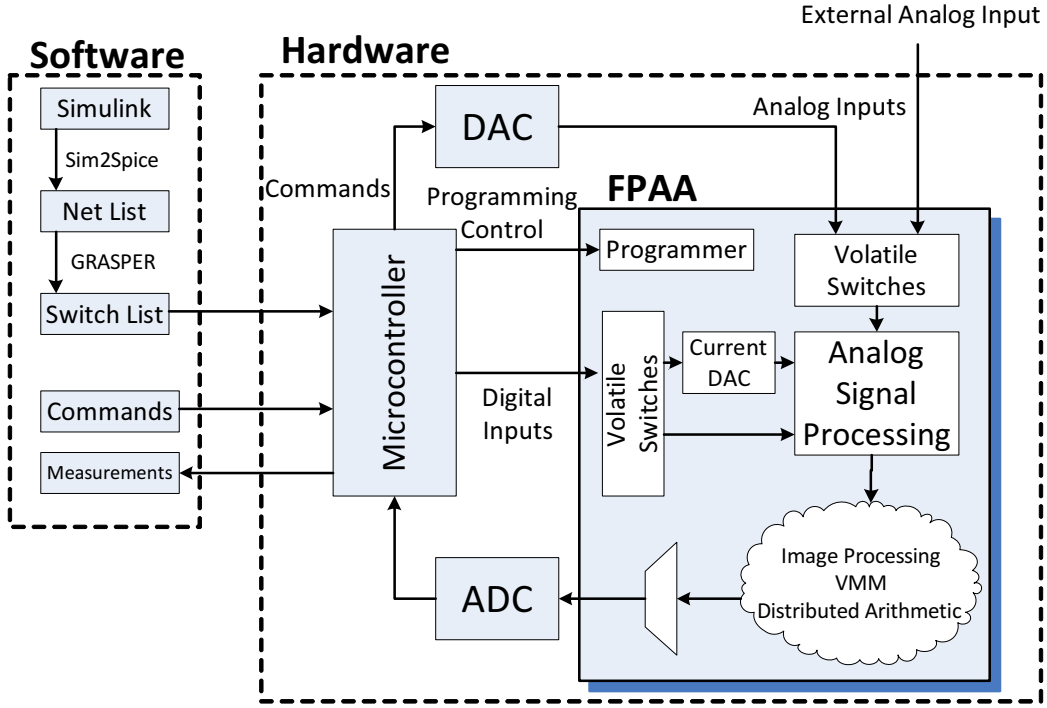


Figure 16: Signal flow in the RASP 2.9v.

The hardware consists of a test board, which houses a microcontroller, DAC, ADC, and the FPAA. The floating-gate targets from GRASPER are programmed using an on-chip programmer, with programming commands sent from the on-board microcontroller. Analog inputs can come into the FPAA from DACs located on the test board or from the external environment. These analog signals are processed using CABs very similar to those of the RASP 2.8a. Analog outputs are sent to an on-board DAC and relayed back to the microcontroller.

The above features are by now standard for FPAAs developed by the CADSP lab. The RASP 2.9v has a number of major characteristics that distinguish it from previous versions. First, as noted in Fig. 16, digital and analog inputs can be applied to the chip through volatile switches. These switches consist of digital shift registers attached to transmission gates. One side of the transmission gate is tied to a common

line, while the other is attached to nets in the switch matrix. This is illustrated in Fig. 17. Inputs can be applied to the switch matrix by activating the T-gate corresponding to the desired net and applying a signal to the common line. Outputs can be muxed from the switch matrix using these volatile switches.

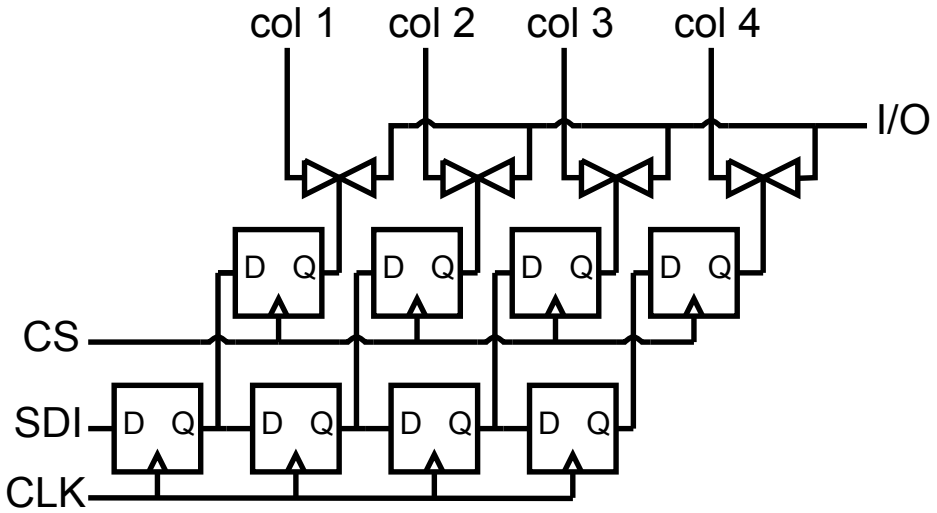


Figure 17: Architecture of the volatile switches in the RASP 2.9v.

As mentioned earlier, the purpose of this chip is to facilitate communication between the analog and digital signal processing worlds. One canonical digital operation is vector-matrix multiplication (VMM). The CADSP lab has developed a reliable method for performing VMM using reconfigurable architectures [20]. The RASP 2.9v is specifically designed to facilitate VMM. The first design choice was to add CABs which are well-suited to compiling VMMs. These CABs have a large number of OTAs, and there are local connections which allow very few switches to be programmed in order to create a VMM.

A second important characteristic of the RASP 2.9v which enables VMM is its hybrid switch matrix, meaning that some of its floating-gate elements are programmed using a direct scheme, and others are programmed using an indirect scheme. Direct programming means that one FG-pFET device is both programmed and used in run-time circuits. Indirect programming means that one device is programmed,

but a different device with the same gate voltage is used in run-time circuits. The reason indirect programming is performed is that, in FPAA switch matrices, during program-mode devices must be isolated using a series pFET switch. In run-mode, this switch cannot pass low voltages. So the floating-gate switch works poorly in directly-programmed structures. In indirect cases, there is no switch isolating the run-mode transistor. This is very simply illustrated in Fig. 18.

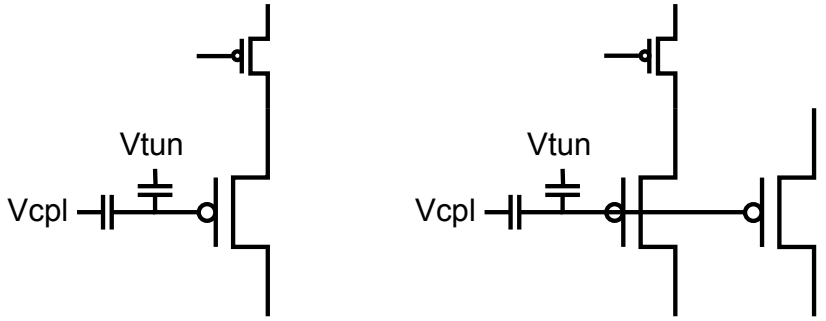


Figure 18: Indirect vs. Direct programming in the RASP 2.9v

The disadvantage to using indirect programming is that there is inherently mismatch between the programmed device and the run-mode devices. This mismatch is a big problem for current-mode circuits such as the VMM. The two devices have the exact same floating-gate voltage, so any mismatch in their physical characteristics (most notably threshold voltage) results in different currents through the two devices for the same set of terminal voltages. Direct programming solves this problem. Current-mode circuits don't need to conduct all the way to ground, so direct programming is ideal since the programmed device is the same device used in run-mode, so mismatch is eliminated.

The other major feature of the RASP 2.9v is that it contains CABs which are well-suited for compiling current-mode DACs. A current-mode DAC is ideal for this chip because the VMMs are also current-mode, meaning that digital signals can be sent from the DAC to the VMM without any intermediate V/I conversions. Details of these DACs are discussed in the following section.

3.2 Design of the DAC CAB

A representative schematic of the DAC CAB is shown in Fig. 19. The architecture of the CAB itself is relatively simple. A pFET differential pair steers current between a common output node and ground. The input currents to the pair I_{in_i} come from the switch matrix, external to the CAB. This means that the currents can be generated by an on-chip or off-chip source and routed to the CAB through the switch fabric. However, the CAB was designed in such a way that input currents could be generated from floating-gate pFETs in the switch matrix itself, which makes a very compact structure.

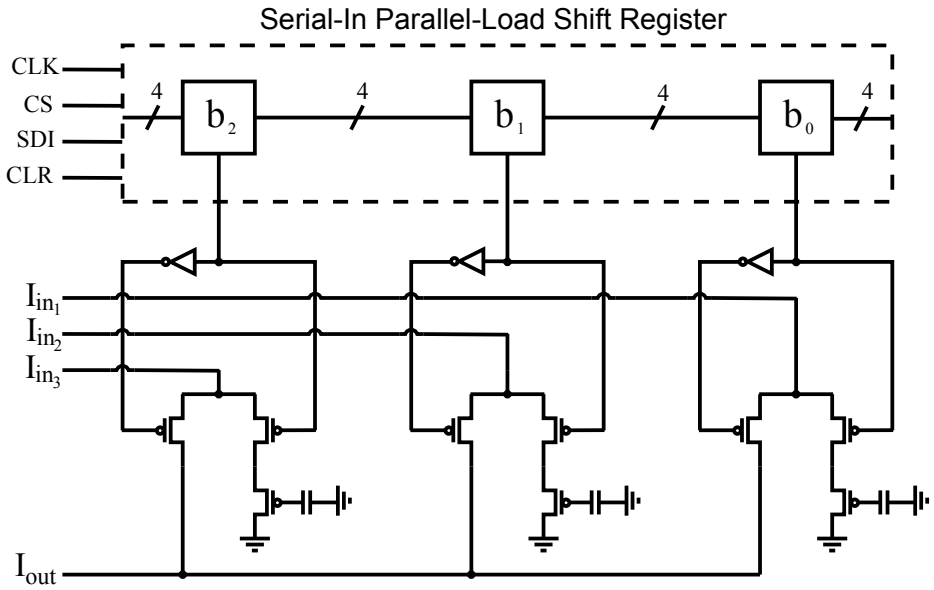


Figure 19: Schematic for three bits of the 8-bit DAC CAB.

The signal which determines whether a particular current is added to the total output or not comes from a serial-in parallel-load shift register. We used the same type of shift register as used in the volatile switches. The chip-select signal, which loads data from the input to the output register, is determined by a simple addressing scheme. We have multiplexed the SDI signal so that it can either come from the global SDI line or from a column of the switch matrix. This allows for the DAC to take data from other systems on the chip, increasing the types of systems we can implement

with the DAC.

While this CAB could have a number of uses due to its reconfigurable nature, it was designed with two particular DAC architectures in mind. The first was a binary-weighted current-steering DAC. A schematic example of this type is shown in Fig. 20. As mentioned above, the binary-weighted current sources would be floating-gate current sources from the switch matrix. These can be trimmed to very precise values with floating-gate programming. This addresses one of the dominant problems in subthreshold current-mode DACs: offset. Any small mismatch between two current sources can have huge effects on the DAC's output. Adding in programmability greatly reduces the offset.

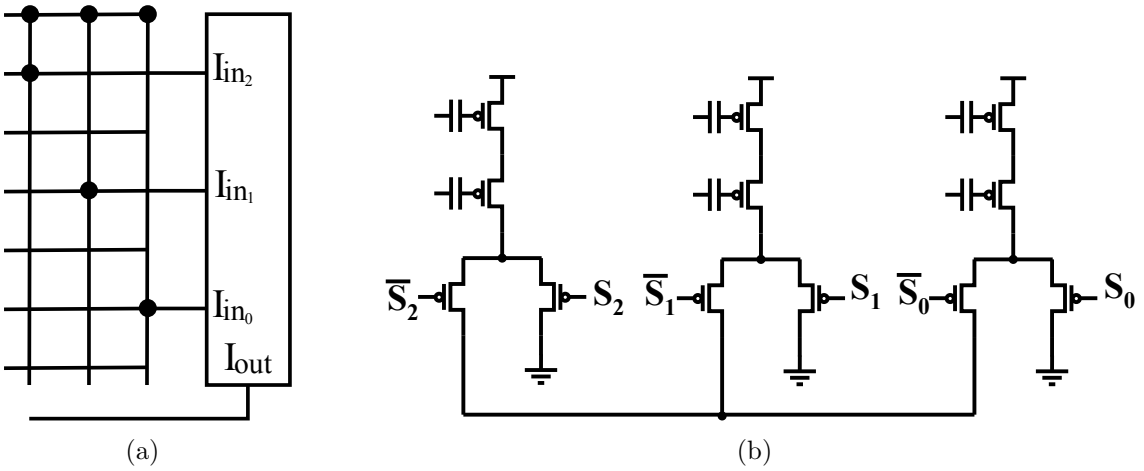
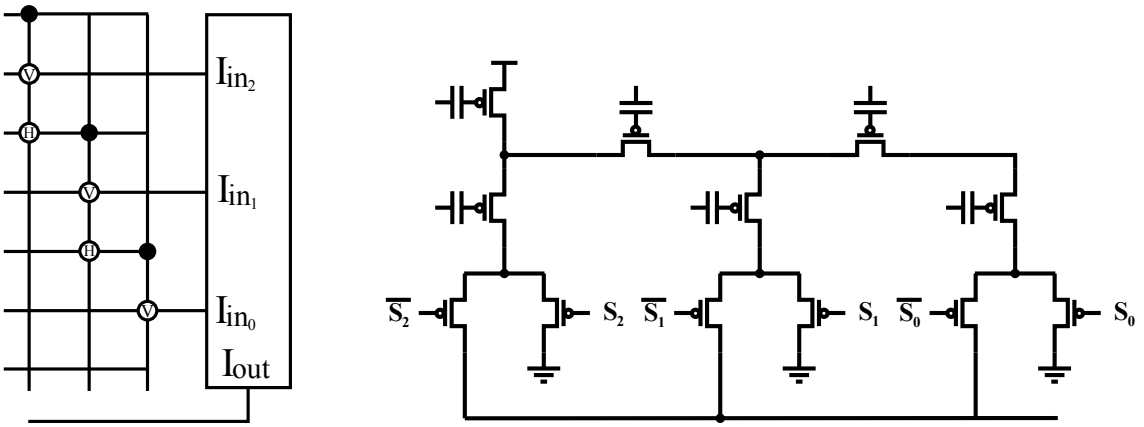


Figure 20: (a) Switch settings for compiling a binary-weighted DAC. (b) Resulting schematic when switches from (a) are programmed.

A second type of DAC which was designed to be compiled on this architecture is an R-2R DAC, or diffusor structure. A schematic example of this is shown in Fig. 21. A single floating-gate input current is applied to the first node of the structure. The current is then divided by a proportion set by the floating-gate programming of the longitudinal and leakage pFETs. This current is then used as a binary (or some other proportion) weight for the DAC.



(a) (b)

Figure 21: (a) Switch settings for compiling an R2R DAC. (b) Resulting schematic when switches from (a) are programmed.

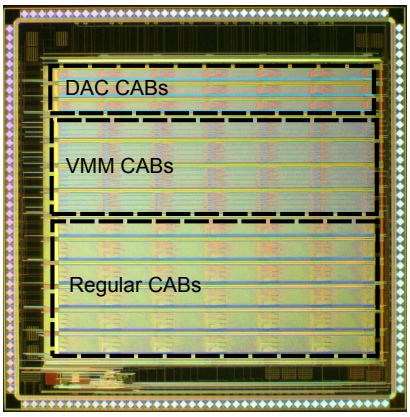
Table 2: RASP 2.9v Specifications

Spec	Value
Process	350nm CMOS
Die Size	5mm x 5mm
CABs	18 DAC, 36 Regular, 24 VMM (x4 input structures)
Chip I/O	79 Analog, 20 Volatile lines
Number of Volatile Switches	4728: 6 x 400-bit (vertical) 14 x 156-bit (horizontal) 6 x 24-bit (DAC)

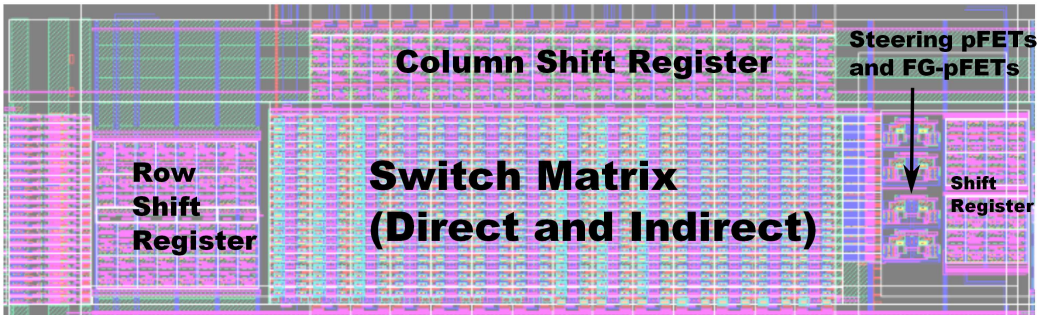
3.3 Experimental Results

The RASP 2.9v was fabricated in a 0.35 μm CMOS process. The resulting die photo is shown in Fig. 22(a), and a representative portion of the layout for which this work was responsible is shown in Fig. 22(b). Some specifications for the chip are given in Table 2.

An 8-bit version of the regular DAC was implemented. In order to achieve higher accuracy, we used a heuristic to trim the floating-gate targets. We give the FPAA initial targets for each of the bits, measure the actual current through the bits, and then modify the targets to get closer to the ideal behavior. This heuristic is shown in pseudocode below.



(a)



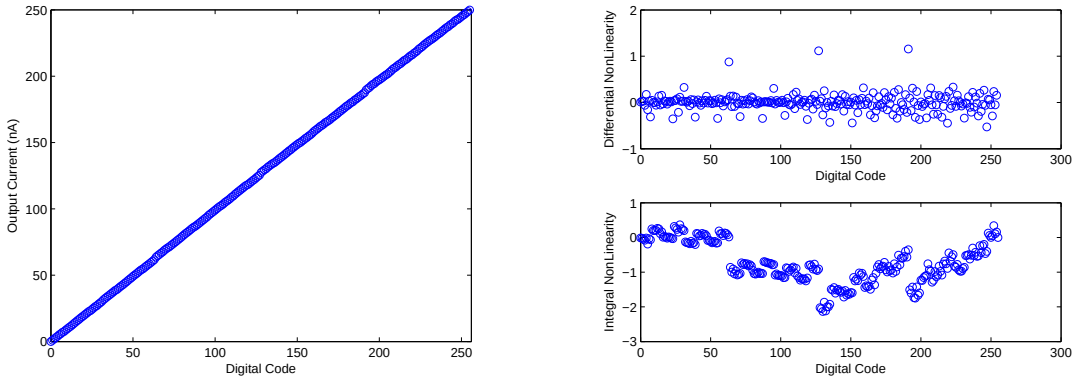
(b)

Figure 22: (a) Die photo of the RASP 2.9v chip. (b) Detailed layout picture of the DAC CAB and associated circuitry implemented in this work.

```
I_ideal = LSB*[1 2 4 8 16 32 64 128] % Goal currents
I_prog = I_ideal                     % Initialize programming targets
for i=1:10 {                         % Ten iterations of heuristic
   eraseFPAA()                       % Tunnel and reverse tunnel
   programTargets(I_prog)            % Inject targets for this iteration
   for j=1:length(I_prog) {          % For each target...
      iout = measureBits(j)          % (1) Measure its current
      ratio = I_prog(j)/iout         % (2) Find correction factor
      I_prog(j) = I_prog(j)*ratio }} % (3) Create target for next iteration
```

The results of our initial testing with an 8-bit, 1 nA LSB DAC are shown in Fig. 23. The measured LSB was 0.978 nA, the worst-case INL was -2.13 LSB, and the

worst-case DNL was 1.16 LSB.



(a) (b)

Figure 23: (a) Measured currents for an 8-bit 1-nA LSB DAC (b) DNL and INL for the DAC.

We also compiled an R-2R diffusor structure on the RASP 2.9v. However, the R-2R architecture is much more sensitive to mismatch between floating-gate elements and is difficult to tune because one stage directly depends on the other. Therefore we have not yet achieved results with the R-2R DAC which are as accurate as those from the regular DAC. We hope to improve the tuning heuristic and eventually achieve an R-2R DAC.

Another important part of the DAC CAB architecture is the ability to accept inputs from either the SDI input line or the switch fabric directly. This feature allows for more complex systems to be built, where feedback can be applied from some other subsystem on the chip. We tested this by programming a simple DAC, shifting a series of ones followed by a series of zeros through the SDI, changing the MUX to allow inputs from the switch fabric, and applying the same set of ones and zeros through a digital input from the switch matrix. We got the same response in both cases, indicating that the DACs can take digital input from the switch fabric. This result is shown in Fig. 24.

This work has shown that the DAC CAB functions as it was designed for compiling

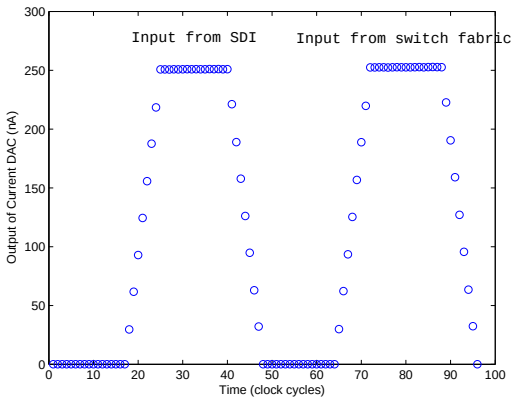


Figure 24: Applying inputs to the DAC both from the SDI input and the switch fabric.

current-steering, binary-weighted DACs. This allowed other work to use the DAC as a building block in more complex systems, such as a small proof-of-concept for an image processing system and mixed-signal FIR distributed arithmetic [21].

CHAPTER IV

UTILIZATION OF FLOATING-GATE PROGRAMMING FOR OFFSET REMOVAL IN A NEUROMORPHIC SYSTEM

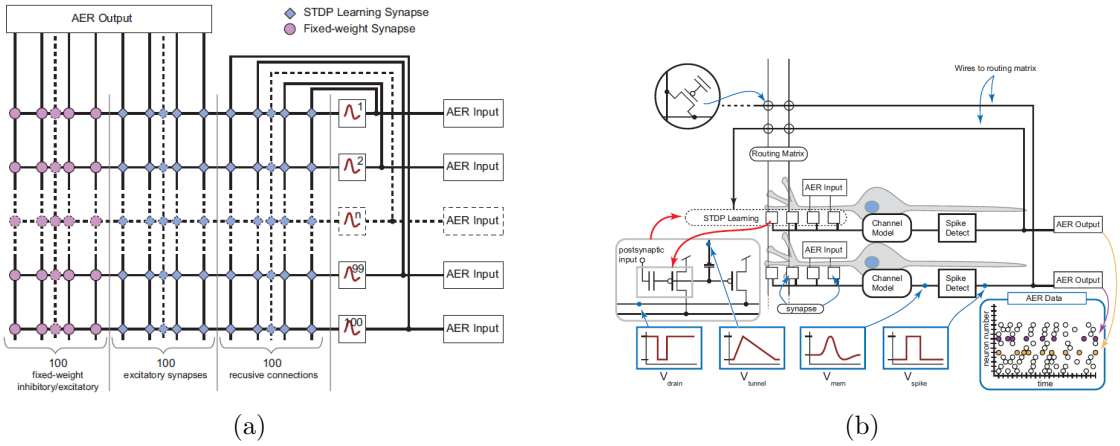
The final project described in this work is the removal of offset in a subsystem of a neuromorphic chip and the development of a software framework for removing offset in other subsystems. Below we will briefly describe the Neuron 1D chip and then present results from the offset removal project.

4.1 Introduction to the RASP Neuron 1D

The RASP Neuron 1D is a reconfigurable neuromorphic chip which builds upon years of neuromorphic research in the CADSP lab. Fig. 25(a) shows the basic architecture of the chip. There are 100 rows of neuron elements, each with 300 synapses (for a total of 30,000 synapses). 200 of those synapses receive inputs from an Address Event Representation (AER) circuit, which is a digital circuit that allows information about action potential "events" to be sent to this chip from another AER-enabled chip or MATLAB. Of these AER synapses, 100 can be inhibitory/excitatory but have a static weight after programming. 100 others are excitatory, and their synapse strengths can change based on a learning rule known as Spike-Time Dependent Plasticity (STDP). Finally, each neuron channel recurrently connects to all of the other neurons, so 100 more excitatory STDP synapses are available.

Fig. 25(b) shows details of how this architecture is implemented. Each synapse is a floating-gate transistor. When an signal is to be transmitted through the synapse, it receives a triangle input waveform at its gate, which causes its output current to look

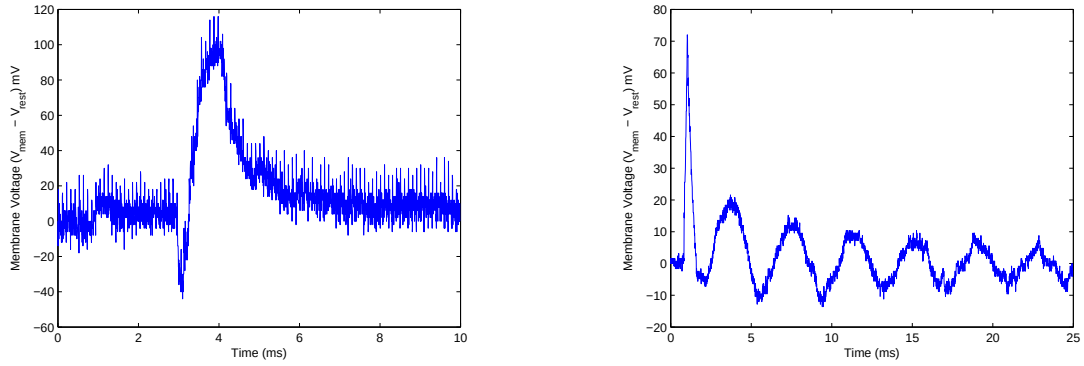
like an Excitatory Postsynaptic Current (EPSC). The charge on the floating-gate determines how strong this current pulse will be. This concept is discussed in [7]. STDP synapses have an extra component which allows them to learn. Their floating-gate charge is modified by a second, indirect transistor. According to the learning rule, the drain and tunneling voltage of this indirect transistor are pulled low and high, respectively. This causes the charge on the indirect transistor to change, and thus the weight of the direct transistor changes. This is discussed in [18].



(a) (b)

Figure 25: (a) High-level overview of Neuron 1D chip. (b) Detailed architecture of biological signals and channels in the Neuron 1D chip. Both images from [3]

The core processing elements of the chip are 100 bio-physically inspired silicon neurons [6]. These differ from the original neurons in that they contain floating-gate transistors to allow trimming of various parameters of the neurons. We typically operate these neurons in an "excitable regime," where a sufficiently strong pulse of current causes the neuron to generate a spike in response. An example of a typical measured response from one of these neurons is given in Fig. 26(a). We have noticed some interesting dynamic properties of these neurons. For example, one measurement showed a neuron undergoing subthreshold oscillations (see Fig. 26(b)), which is generally seen in neurons very near a bifurcation.



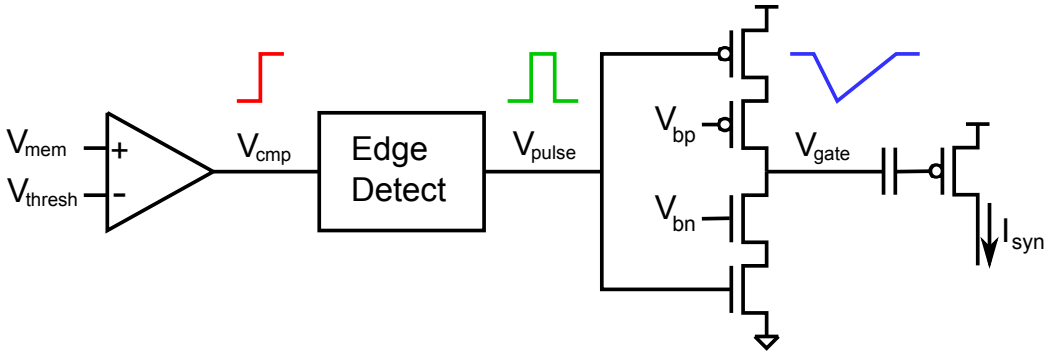
(a) (b)

Figure 26: (a) Typical measurement from excited neuron. (b) Neuron exhibiting subthreshold oscillations.

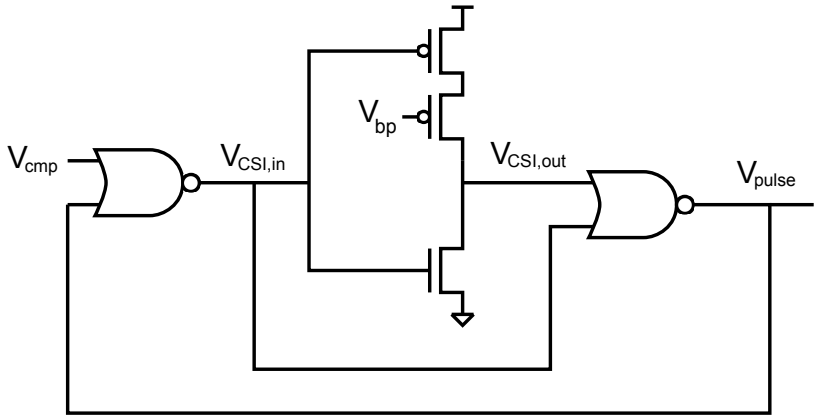
4.2 Offset Removal in the Neuron 1D Gate Waveform Shaping Circuitry

As mentioned before, the EPSC generated by a synapse depends on its floating-gate voltage and the waveform that is applied to its gate. This waveform is generated by gate waveform shaping circuitry, which is shown in Fig. 27. When the membrane potential V_{mem} becomes greater than some user-determined threshold V_{thresh}, a comparator trips. The comparator output goes into an edge detect circuit, which generates a pulse whose length is determined by half of a current-starved inverter. This pulse then goes into a full current-starved inverter. The biases for all three starved current sources are generated by floating-gate pFETs. So the risetime, falltime, and pulsewidth of the system components are all determined by these three floating gates.

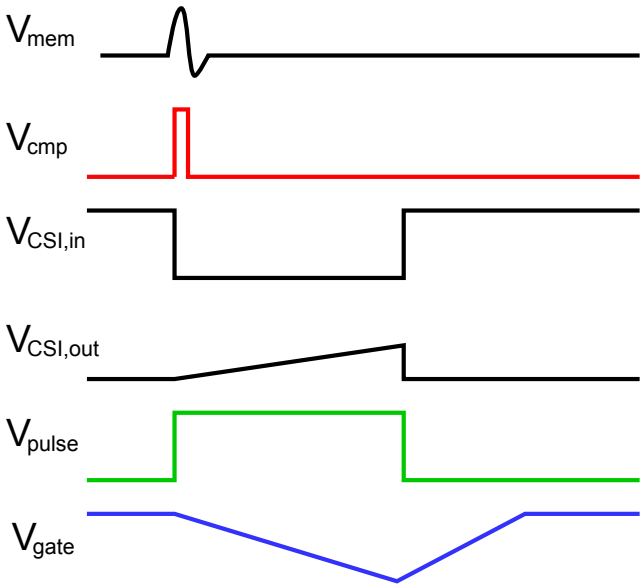
While biology demonstrates a wide degree of variation in its synapses, we would like to set up our system such that this variability can be added if desired. In fact, we may want variability with a certain statistical distribution, so having control over these parameters is crucial. It turns out that the circuits in the gate waveform shaping circuitry are perfect for offset removal because they have a very simple operation. Both the edge detection and the current-starved inverters are based on the slewing



(a)



(b)



(c)

Figure 27: (a) Gate waveform shaping circuitry. (b) Gate-level schematic of edge detect circuitry. (c) Timing diagram of gate waveform shaping.

operation of a capacitor with a fixed current being applied to it:

$$\frac{\partial V}{\partial t} = \frac{I}{C} \tag{16}$$

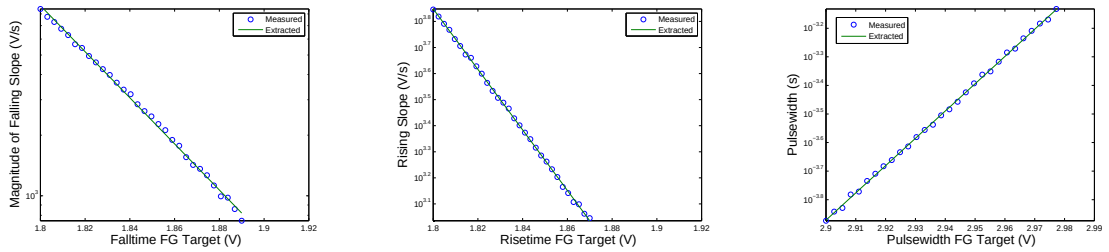
We know that this fixed current has an exponential relationship to the floating-gate voltage of the pFET.

$$I = I_0 e^{\frac{V_s - \kappa V_{fg}}{U_T}} \tag{17}$$

The falltime, risetime, and pulsewidth of the circuitry are all directly proportional to the slew rate of these caps. The slew rates are exponentially dependent on the floating-gate potential. Therefore we can assume that all of these parameters follow the relationship given below:

$$param \propto I_{0_{param}} e^{\frac{\kappa_{eff_{param}} V_{fg}}{U_T}} \tag{18}$$

The V_s term dropped out because it is assumed to be equal to V_{DD} and is incorporated in the $I_{0_{param}}$ term. We verified this exponential dependence for all three parameters, and this is shown in Fig. 28.



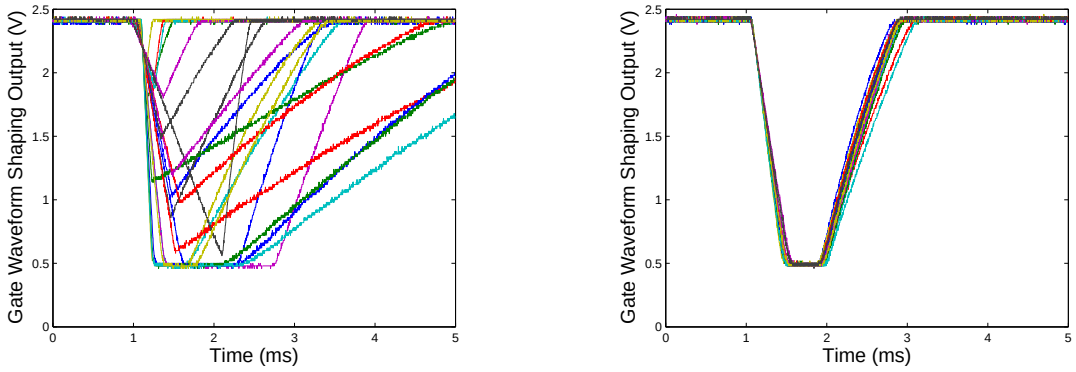
(a) (b) (c)

Figure 28: Dependence of (a) falltime, (b) risetime, and (c) pulsewidth on the floating-gate voltage.

Another assumption we made is that κ does not change very much from pFET to pFET. This means that we can extract κ_{eff} once, and then find $I_{0_{param}}$ for each channel. To find κ_{eff}, we program one channel to different floating-gate voltages

and measure the slope of the resulting response. To find $I_{0_{param}}$ for a channel, we only program it to one floating-gate target, measure the parameter value, and extract what $I_{0_{param}}$ must be from Eq. 18. Once we have κ_{eff} and $I_{0_{param}}$, we can achieve any waveform shaping parameter that we desire.

We have so far done this experiment on a small subset of the gate waveform shaping circuits, and the results are shown in Fig. 29. These results are for the channels which were programmed correctly. The programming infrastructure is still being improved, so sometimes a waveform will not be captured as part of the characterization process. Programming errors are not shown here. As one can see, the agreement between the gate waveform shapes is dramatically improved.



(a) (b)

Figure 29: Output of waveform shaping circuits (a) before and (b) after offset removal.

The statistics of the waveform characteristics have been drastically improved. The histograms of the four waveform characteristics are shown in Fig. 30.

A significant portion of this project was developing a codebase which allows for automatic characterization of these channels. This set of code needed to be compatible with already-existing code for programming neural circuits on the Neuron 1D. An example of how to use what was developed is listed in Appendix A.

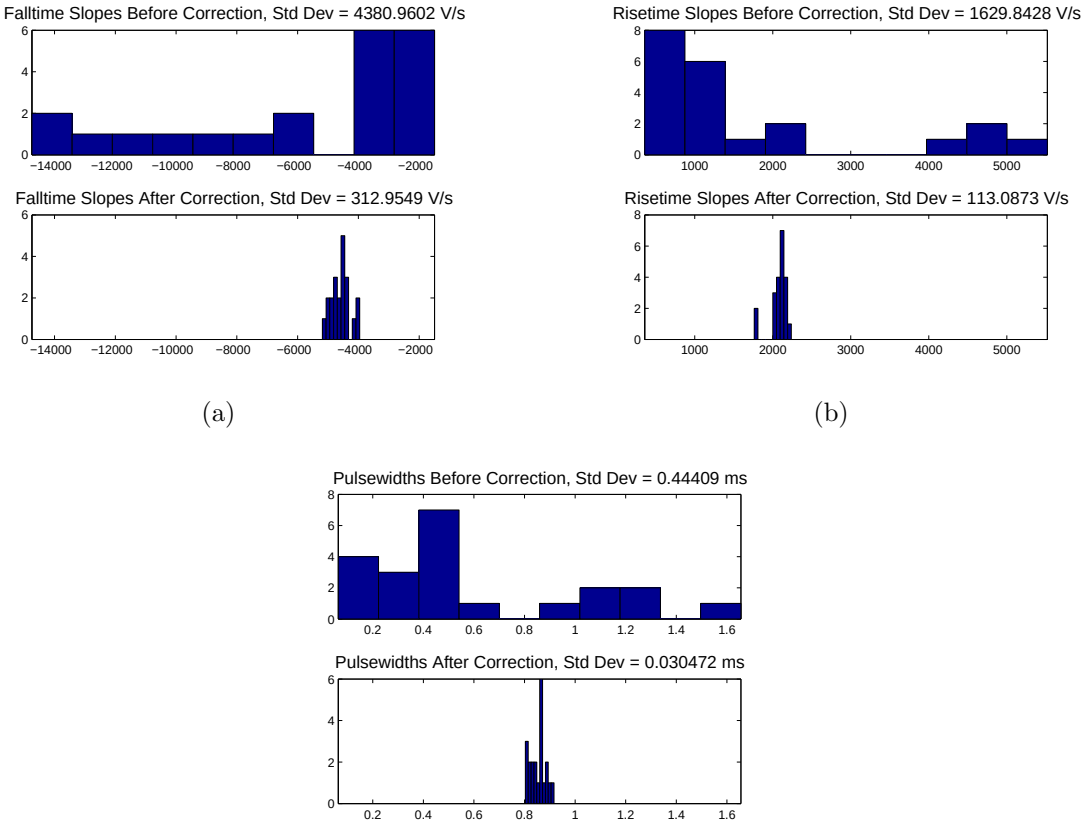


Figure 30: Histograms of (a) falltime, (b) risetime, and (c) pulsewidth before and after offset correction.

CHAPTER V

CONCLUSION

5.1 Summary of Contributions

The primary contribution of this thesis was the implementation and analysis of dendrite circuits on an FPAA. The circuits demonstrated steady-state and dynamic characteristics typical of linear cable models seen in biology. Issues pertaining to non-idealities of the FPAA environment and the operating conditions of the circuits were also discussed. The secondary contribution was the design, layout, and testing of a portion of a new type of FPAA. A current-mode DAC was successfully compiled on the resulting chip. The final contribution was a software routine which automatically characterizes and removes offset in a subcircuit of a neuromorphic system. This will provide experimenters much better control over the behavior of their neural circuits.

This work has generated an accepted journal article [17] and is a component of two other journal articles. One has been submitted for publication [21], and one is to be submitted [3].

5.2 Future Directions

Some of these contributions have immediate short-term consequences for research directions. First, the methodology for removing offsets in the Neuron 1D chip needs to be applied to other subsystems. For example, the floating-gates which behave as synapses will exhibit mismatch from device to device. Coming up with a scheme to quantify and compensate for this mismatch would be very useful. In the past, we were forced to hand-tune floating-gate parameters to get neural circuits such as synfire chains to work correctly. This is entirely infeasible when larger systems are

desired.

In the long-term, the cable properties of dendrites have the potential to add computational complexity to neuromorphic systems. Now that the basic cable properties have been proven, we need to investigate what types of computations dendrites allow us to do in systems with active channels. As stated before, simple primitives such as directional selectivity and spatiotemporal coincidence detection have been shown to be important computations performed be dendrites. These primitives alone could be enough to create interesting computational systems, such as classifiers. We hope to show that remaining faithful to biology increases the amount of low-power, robust computation possible on a CMOS substrate.

There are other possible directions for research which are not direct results of the material presented here, but are closely related. One interesting project would be using neuromorphic models of neuron somas to create analog dynamic clamp systems. This would remove the large amount of equipment needed for current dynamic clamp systems. Another interesting area is the use of these neuromorphic platforms to implement canonical neural computations, such as winner-take-all ring networks, synfire chains, and oscillators. I have also become interested in the nonlinear dynamics which describe how these neural circuits work. The development of a CMOS model similar to the hybrid Izhikevich model is also an interesting problem. Regardless of which direction I eventually choose, I am convinced that this thesis was an invaluable introduction to neuromorphic computing and reconfigurable systems.

APPENDIX A

EXAMPLE CODE FOR NEURON 1 CHIP

```
%% Use CharacterizeNeuron1dChip.m
% In machine dependence, set g.machineParams.chipID = x, where x is an integer

opts = struct; nums = struct; ss = struct;

opts.NeuronGateRef = true; opts.NeuronGateChans = true; opts.AERGateRef = true;
opts.AERGateChans = true;
nums.neuron = 0:3; nums.aer = 0:3;  % Channels to  characterize
ss.instname = 'SCOPE1'; ss.chan = 2; ss.nsamples = 2499; %Scope settings
refChan = 50;  %The "reference" channel will be #50
Nref = 10;   % Number of points to be used to characterize reference channe

tic;
CharacterizeNeuron1dChip(opts, nums, refChan, Nref, ss);
tchar = toc;

display(['Characterization of ' num2str(length(nums.neuron)) ' neurons and '
num2str(length(nums.aer)) ' chans took ' num2str(tchar) ' seconds'])

%% See how well characterization worked.
% First, specify the architecture

% Load characterization data
tic
load([g.machineParams.runDir '\Matlab code\hardware specific\Neuron1d\
ChipChars\Chip' num2str(g.machineParams.chipID) '.mat'])
neuronchars = neuron; aerchars = aer;
```

```
%Specify neuron topology
a = pynnSession;
x = a.create('HaslerNeuron',{},length(nums.neuron));
in = a.create('SpikeSourceArray',{},length(nums.aer));
for i=1:length(nums.neuron)
    set(x(i),{'placement' nums.neuron(i)});
end
for i=1:length(nums.aer)
    set(in(i),{'placement' nums.aer(i)});
end
set(x,{'useGateWvfm' 1});
set(in,{'spike_times' 50});
a.compile

% Copy characterization data into resulting structs
neuron = a.targets_output.neuron;
aer = a.targets_output.aer;

for i=1:length(neuron)
    neuron(i).chars = neuronchars(i).chars;
end
for i=1:length(aer)
    aer(i).chars = aerchars(i).chars;
end

% Remove offsets
[neuron,aer] = ParamsToTargets(neuron,aer);

%% Just program relevant FGs to test the basic idea

% Recurrent Measurements
a = [neuron.targets];
```

```
rcr_fts = [a.rcr_ft];
rcr_rts = [a.rcr_rt];
rcr_pws = [a.rcr_pw];

[t_rcr, vouts_rcr, fail_rcr, err_rcr] =
Neuron1dCharFns.CaptureRCRWaveforms(nums.neuron, rcr_fts, rcr_rts, rcr_pws, ss)

% AER Measurements
b = [aer.targets];
aer_fts = [b.aer_ft];
aer_rts = [b.aer_rt];
aer_pws = [b.aer_pw];

[t_aer, vouts_aer, fail_aer, err_aer] =
Neuron1dCharFns.CaptureAERWaveforms(nums.aer, aer_fts, aer_rts, aer_pws, ss)



% Plot the results
figure(1)
plot(t_rcr,vouts_rcr')
title('Recurrent Gate Waveforms')
xlabel('Time (s)')
ylabel('Gate Wvfm Voltage (V)')

figure(2)
plot(t_aer,vouts_aer')
title('AER Gate Waveforms')
xlabel('Time (s)')
ylabel('Gate Wvfm Voltage (V)')

%% Now test the entire programming infrasructure

% Program just the timing FGs
```

```
Neuron1dFns.tunnel('all'); Neuron1dFns.reverseTunnel('all');
unused.leak=true; unused.comparator=true;
unused.extra_gate=true; unused.stdp=true;
tun.timing=1; tun.stdp=1; tun.nonstdp=1; tun.chans=1; tun.ota=1;
[err] = VectorizedProgramming(neuron,aer,unused,tun);
tprog = toc;
display(['Done programming characterization test in
 ' num2str(tprog) ' seconds'])


% Now measure waveform characteristics
[t_rcr, vouts_rcr, fail_rcr] =
Neuron1dCharFns.CaptureProgRCRWaveforms(nums.neuron, ss);
[t_aer, vouts_aer, fail_aer] =
Neuron1dCharFns.CaptureProgAERWaveforms(nums.aer, ss);

figure(1)
plot(t_rcr,vouts_rcr')
figure(2)
plot(t_aer,vouts_aer')
```

REFERENCES

[1] Basu, A., Brink, S., Schlottmann, C., Ramakrishnan, S., Petre, C., Koziol, S., Baskaya, F., Twigg, C., and Hasler, P., "A floating-gate-based field programmable analog array," *IEEE Journal of Solid-State Circuits*, vol. 45, pp. 1781–1794, 2010.

[2] Branco, T., Clark, B. A., and Hausser, M., "Dendritic discrimination of temporal input sequences in cortical neurons," *Science*, vol. 329, pp. 1671–1675, September 2010.

[3] Brink, S., Ramakrishnan, S., Nease, S., Shapero, S., Wunderlich, R., Hasler, P., Basu, A., and Degnan, B., "A learning-enabled neuron array ic based upon transistor channel models of biological phenomena," *IEEE Transactions on Biomedical Circuits and Systems (To be Submitted for Publication)*.

[4] Destexhe, A., "Dendrites do it in sequences," *Science*, vol. 329, pp. 1611–1612, September 2010.

[5] Elias, J. G., "Artificial dendritic trees," *Neural Computation*, vol. 5, pp. 648 – 663, 1993.

[6] Farquhar, E. and Hasler, P., "A bio-physically inspired silicon neuron," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 52, pp. 477 – 488, 2005.

[7] Gordon, C., Farquhar, E., and Hasler, P., "A family of floating-gate adapting synapses based upon transistor channel models," *Proceedings of the 2004 International Symposium on Circuits and Systems, 2004. ISCAS '04.*, vol. 1, pp. 317 – 320, 2004.

[8] Hasler, P., Koziol, S., Farquhar, E., and Basu, A., "Transistor channel dendrites implementing hmm classifiers," *Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium on*, pp. 3359 – 3362, 2007.

[9] Hille, B., *Ion Channels of Excitable Membranes, Third Edition*. Sunderland, MA: Sinauer Associates, Inc, 2001.

[10] Koch, C., *Biophysics of Computation*. New York, NY: Oxford University Press, 1999.

[11] Koch, C. and Segev, I., "The role of single neurons in information processing," *Nature Neuroscience*, vol. 3, pp. 1171–1177, Nov. 2000.

[12] Koziol, S., Schlottmann, C., Basu, A., Brink, S., Petre, C., Degnan, B., Ramakrishnan, S., Hasler, P., and Balavoine, A., "Hardware and software infrastructure for a family of floating-gate based fpaas," *Circuits and Systems, 2010. ISCAS 2010. IEEE International Symposium on*, pp. 2794 – 2797, 2010.

[13] Lee, E. K. F. and Hui, W. L., "A novel switched-capacitor based field-programmable analog array architecture," *Analog Integrated Circuits and Signal Processing*, vol. 17, no. 1-2, pp. 35–50, 1998.

[14] Liu, S.-C., Kramer, J., Indiveri, G., Delbruck, T., and Douglas, R., *Analog VLSI: Circuits and Principles*. Cambridge, MA: The MIT Press, 2002.

[15] London, M. and Hausser, M., "Dendritic computation," *Annual Review of Neuroscience*, vol. 28, pp. 503–532, July 2005.

[16] Mead, C., *Analog VLSI and Neural Systems*. Reading, MA: Addison-Wesley, 1989.

[17] Nease, S., George, S., Hasler, P., Koziol, S., and Brink, S., "Modeling and implementation of voltage-mode cmos dendrites on a reconfigurable analog platform," *IEEE Transactions on Biomedical Circuits and Systems (Accepted for Publication)*.

[18] Ramakrishnan, S., Hasler, P., and Gordon, C., "Floating gate synapses with spike-time-dependent plasticity," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 5, pp. 244 – 252, 2011.

[19] Rasche, C. and Douglas, R. J., "Forward- and backpropagation in a silicon dendrite," *IEEE Transactions on Neural Networks*, vol. 12, pp. 386 – 393, Mar. 2001.

[20] Schlottmann, C., Petre, C., and Hasler, P., "Vector matrix multiplier on field programmable analog array," *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, pp. 1522 – 1525, 2010.

[21] Schlottmann, C., Shapero, S., Nease, S., and Hasler, P., "A digitally enhanced dynamically reconfigurable analog platform for low-power signal processing," *Journal of Solid-State Circuits (Submitted for Publication)*.

[22] Segev, I., Rinzel, J., and Shepherd, G. M., *The Theoretical Foundation of Dendritic Function: Selected Papers of Wilfrid Rall with Commentaries*. Cambridge, MA: The MIT Press, 1995.

[23] Shapero, S. and Hasler, P., "Precise programming and mismatch compensation for low power analog computation on an fpaa," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, submitted for review.

[24] Twigg, C., Gray, J., and Hasler, P., "Programmable floating gate fpaa switches are not dead weight," *Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium on*, pp. 169–172, May 2007.

[25] Wang, Y. and Liu, S.-C., "Input evoked nonlinearities in silicon dendritic circuits," *IEEE International Symposium on Circuits and Systems*, pp. 2894 – 2897, 2009.